Capstone Project Document

Automatic School Bell System Project

ASBS	
Group Members	Pham Nam Thang - 60495 Nguyen Huu Tai – 60042 Nguyen Xuan Truong - 60465 Le Viet Anh – SE02219
Supervisor	Hoang Xuan Son

- Hanoi, 06/2015 -

Table of Contents

I. Introduction 1
1. Team member 1
2. The initial idea 1
3. Name of this capstone project 2
4. Existing system 2
4.1. Ampletrails Automatic School Bell | Model ASBAO 2
4.2. BellMinder 4
4.3. ASBS's advantages 5
5. System overview description 5
6. Scope 7
II. Project Management Plan (PMP) 8
1. Project Organization 8
1.1. System process model 8
1.2. Role and responsibility 9
1.3. Tool and Techniques 11
2. Project Management Plan 14
2.1. Tasks 14
2.2. Task Sheet: Assignment and timetable 15
2.3. Coding convention 15
2.4. Dependencies and Constrains 16
3. Risk 16
III. System Requirement Specification (SRS) 18
1. User Requirement Specification 18
1.1. Control bell system by using mobile application 18
1.2. Control bell system by using PC 19
2. System Requirement Specification 19
2.1. External Interface Requirement 19
2.2. Functional requirement 20
2.3. Non-functional Requirement 45
2.4. Prototype 46
IV. System Design Document (SDD) 57
1. System Overview 57
2. System Architecture 58
2.1. Overall System Architecture 58
2.2. Mobile Application Architecture 59

3. Detail Design 61
3.1. Control bell system 61
4. User Interface Design 82
5. Hardware Interface Design 82
5.1. Hardware description 82
5.2. Hardware component 85
5.3. Communication protocol – ASBS Command 86
6. Data Structure 89
6.1. Database structure design 89
V. System Implement & Test 89
1. Test Approach 90
2. Screen Shots 93
3. Test Plan 93
3.1. Feature to be tested 93
3.2. Feature not to be tested 93
3.3. Test strategy 93
4. Test Case 95
5. Other material 95
5.1. UI Test result 95
5.2. Unit test firmware result 95
5.3. System test result 95
VI. Conclution 97
VII. System User's Manual (SUM) 98
1. Android Application 98
1.1. Detect and connect to bell system 98
1.2. Turn on bell 99
1.3. Update schedule 100
1.4. Change authentication password 104
2. PC Application 106
2.1. Detect and connect to bell system 106
2.2. Turn on bell 107
2.3. Update schedule 108
2.4. Setting 109
2.5. Status 110

I. Introduction

In this section:

- Provides starting information of the project like team member (in section 1.) and the initial idea (in section 2.).
- Comments existing systems which have same function with their advantages, disadvantages and the advantages of ASBS compare to them (in section 4.)
- Pays particular attention to the system overview (in section 5.); describe system's functions and the link among them.
- Outline the scope of the project's scope (in section 6.)

1. Team member

No	Name	Roll number	Role	Email Contact
1	Pham Nam Thang	60495		thangpn60495@fpt.edu.vn
2	Nguyen Huu Tai	60042		tainh60042@fpt.edu.vn
3	Nguyen Xuan Truong	60465		truongnx60465@fpt.edu.vn
4	Le Viet Anh	SE02219		anhlvSE02219@fpt.edu.vn

2. The initial idea

Automatic School Bell System abbreviated as ASBS is an automatic system for queuing in any organizations such as School, Banks, Hospitals, Service Companies, Universities, Building etc. It is not newest version in this kind of product but quite suitable for some normal and small organizations which need a useful and cheap system. Actually, we develop this for our school, FPT University where has many problems in servicing and consulting for students' or others' questions. It help FPT University Staff easy to control dormitory and auditorium.

In our thinking, ASBS is very useful for FPT University or other organizations. It is not only necessary for staffs who servicing directly, but also for whole stakeholders who related to this system because it will bring staffs comfortable in announcement, convenience for customers and more professional for that firm really. Therefore, we aim to develop this system to improve work effect for companies and satisfaction from customers.

In the market today, there are many product such as Automatic School Bell Systems. But it too old and not suitable University school. In this scope of our

project, we just develop some useful and necessary functions that suitable for FPT University.

Come from actual needs and suggestion of our supervisor, Mr. Son, our group decided to develop a project provide Automatic School Bell System aim to resolve problems related to priority of customers who need to make notification between them and another for examples: Hospitals, Schools, Banks, Stations, etc with more satisfactions and save time, cost for both consumers and organizations as well.

ASBS is developed with 4 main modules in original. The first module is abstracted in Android device to control all other modules' operations and also interacts with users in software programmed in Android framework by using Java. In second module, we designed a small main controll broad includes Raspberry Pi (adruino) and NRF24L01 to make communication with other modules in system. The third module will display to customers all important information in a 7-SEG-LED. And the last module is ringing system.

3. Name of this capstone project

The official name of this project is ASBS – Automatic School Bell System Project.

4. Existing system

4.1. Ampletrails Automatic School Bell | Model ASBAO

Beta Technologies introduce Automatic School Bells for the first time in India. The school bells are designed with most reliable Micro Controller based technology. The school bell system with good accuracy and unique features includes individual time settings from Sunday to Saturday timing, Exam, Weekend Timing, etc. It can be installed in Schools, Colleges, Mills and also in Industries.

Figure 1: Basic Ampletrails Automatic School Bell

Features:

- PC Interfacing with High Efficiency Design
- Two Time Schedules Available
- Each Schedules have individual 24 Time slots
- Each Time Slots can be assigned to have specific time durations
- LCD Display Indication for Day, Date, Time and Next Bell Time
- LED Indication for Schedule Status
- Technologically advanced overall design allows the machine to perform at its best
- This system has provided for individual keys for all application

Auto and Manual modes are available in our system

- Compact and attractive panel for Easy Wall Mounting Installation

Figure 2: System control of Ampletrails Automatic School Bell

4.2. BellMinder

BellMinder is a fully automatic school bell system that is used to solve the common daily problem of ringing a bell on time every time and only when required. Unlike other School Bell Systems on the market, BellMinder is a custom developed School Bell System specifically designed for automating school bells, from the most basic schedules to the most complex. A full year time schedule can be implemented in minutes and schedule changes in seconds. BellMinder has a unique School Bell Program that's based around a clear, easy to follow year planner format. It enables you to assign multiple reusable schedules, using your office PC, by days, weeks or months. Whether your school operates a manual system, audio based, industrial electronic timer or is controlled by your Alarm company, BellMinder beats them all with accuracy, functionality, ease of setup, flexibility and running costs. You can even activate our 'School Lockdown' feature to respond to emergency incidents that can occur in modern schools.

Some advantage of BellMinder: With BellMinder you are no longer limited to just daily timetable schedules. Now you can configure in sports day activations, exam schedules and parent teacher interviews. In fact any event that you can think of can be programmed in seconds.

- With BellMinder you can implement alternative schedules that run in the background and can be called into service at the press of a button.
- These are especially useful if you run Snow Day timetable or a wet day time table. No need to remember to switch back to the main schedule for the next day - BellMinder automatically takes care of that for you, BellMinder has an emergency situation alert feature.
- Automatically night ringing savings and Leap Year aware You can even print hardcopies for school and staff notice boards.
- This School Bell Timer can operate independently of the PC once programmed.
- BellMinder can maintain schedules for many days during a power loss.

4.3. ASBS's advantages

Some advantages of ASBS:

- Support multi-language, especially Vietnamese.
- Connect with pc/ Android
- Control bell in android app.
- Compact system.
- Low price.
- Localize for FPT University.

5. System overview description

ASBS is state-of-the-art solution that helps users build a belling environment which suitable for users, allows users to control Bell system in simple and convenient way with Wi-Fi connection between ASBS system and mobile application on smart phone.

Based on some existed Automatic School Bell System, we develop this system with functions:

- **Bell system**:
    - **Multi-ringing bell**: A system of bell locate around FPT University could be ringing together or ringing individual. The manager could setting it by Android Application on their smart-phone. Not only one bell but also this system allow manager could be control a group of bell. Each bell connect to main controller by RF Transceiver module.
    - **TCP server:** Arduino play a role as a TCP server that handles TCP connection from mobile clients and responses to them.
    - **Fire warning**: This system connect with gas sensor and temperature sensor, too. It's using for detect fire and warning to every one on the building.

- **Mobile application**:
    - **Schedule bell ringing**: Allow users to select their favorite task and scheduling ringing plan of belling system with some action.
    - **Turn on/off bell ringing:** In another side, the system manager should control bells by start or stop bell ringing immediately. Allow users to turn the bell to on or off intermediate.
    - **System notification**: Users can be had the notification about ringing system. Example: the android application will have the ringing task for warning system manage (fire, smoke, move).
    - **Show schedule information on Android**: Prepare an Android application for manage ringing schedule. This application showing the schedule and setting up new task, delete or edit task.

- **PC application**:
    - **Schedule bell ringing**: Allow users to select their favorite task and scheduling ringing plan of belling system with some action.
    - **Setting :** User can update realtime , change password
    - **Bells :** Show all bells , user can add

- **Wi-Fi router**:

- **Wireless LAN provider:** The Wi-Fi router plays a role as a bridge which provide wireless local network (WLAN) for connection between mobile application and Belling system.

6. Scope

Because of Capstone project's scope, we don't focus in building huge system like n Bell connect to n mobile devices and supporting multiple mobile platforms. However, we focus on building stable 1-1 system – 2 Bell connects to 1 mobile device and 2 sub-systems with all functions and features will be constructed and developed.

In this project, we use Arduino platform for building belling system and Android platform for programming mobile application, and optimal control on PC application.

II. Project Management Plan (PMP)

In this section:

- Outlining the project's organization. In this section, the information about chosen system process model will be provided; a detailed outline about team members, their role and responsibility will be given (in section 1.).
- Analyzing all sectors in project management plan including: task, time-line, coding convention, dependencies and constraints, risk (in section 2.).
- Meeting minute collection will be showed as an appendix (in APPENDIX).

1. Project Organization

1.1. System process model

Although almost member have at least 3 years with Java, 1 year with Android Platform and 1 year freelancers experience. But cause of lacking experience in embedded project, ASBS's development team decided to choose interactive and incremental development model.

(The Rational Edge 2003, the Ten Essential of RUP)

1.2. Role and responsibility

1.2.1. Project members

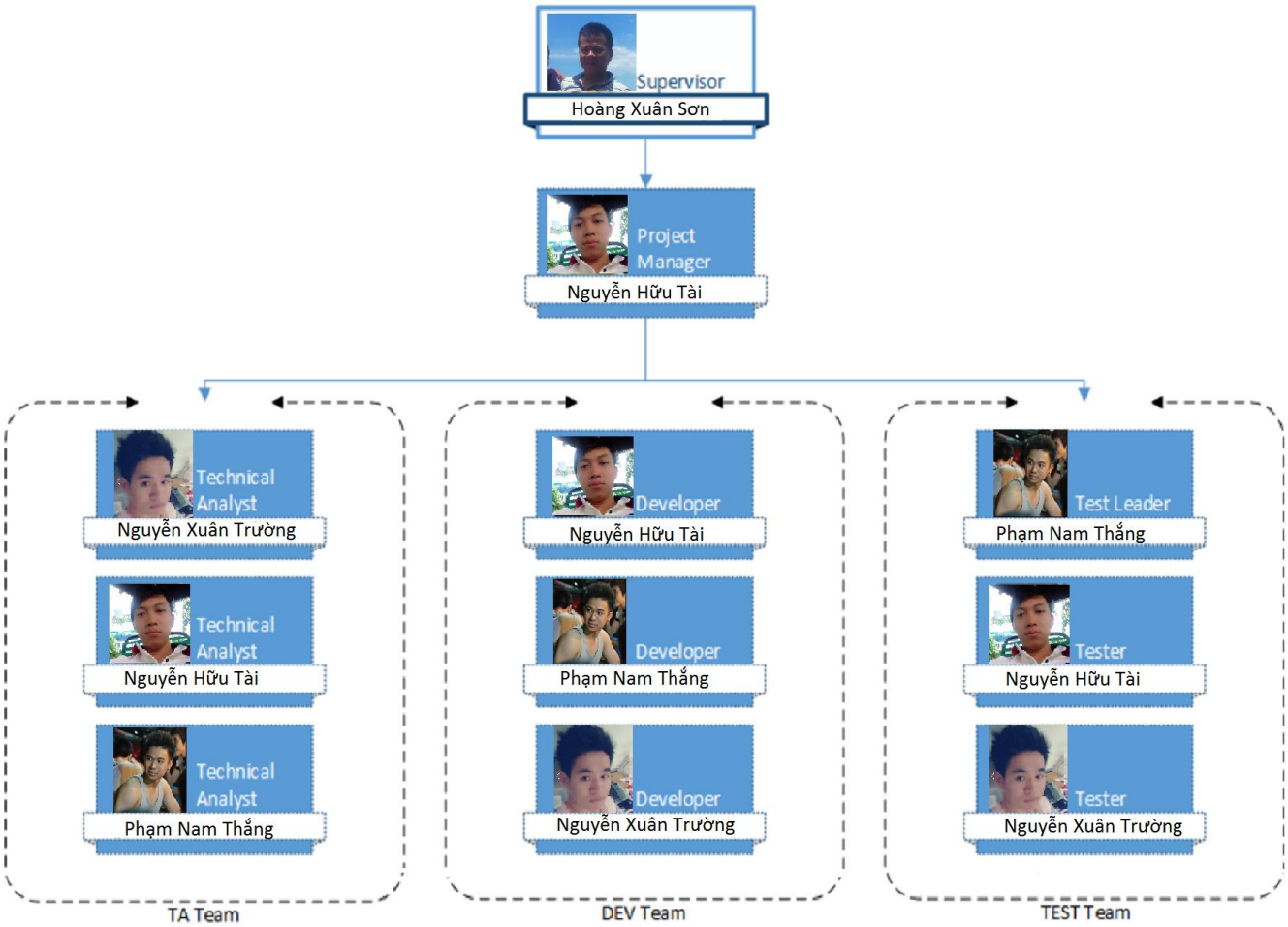


Table 1: Project members

1.2.2. Role and responsibilities

***Project Manager – Nguyen Huu Tai - 60042**: Project Manager is responsible for the day-to-day management of the project life cycle.

- Project Integration Management:
    - Develop project Management Plan, direct and manage project execution, monitor and control project work, close project phase and overall project.
- Project Scope Management:
    - Collect Requirement, define scope, create WBS, verify scope, and control scope.
- Project Time Management:

    - Define activities, sequence activities, estimate activities, estimate human resource, control schedule.
- Project Quality Management:
    - Plan quality, perform quality assurance, and control quality.
- Project Human Resource Management:
    - Manage project team.
- Project Communication Management:
    - Identify stakeholders, plan communication, and distribute information; report performance.
- Project Risk Management:
    - Plan risk management, identify risks, plan risk response, monitor and control risks.

***Project Technical Leader – Pham Nam Thang – 60495**: Project Technical Leader is responsible for technical aspect in all project life cycle.

- Control project technical aspect.
- Research and answer technical questions of team members.
- Report to manager about techniques.
- Create guideline for team members.

***QA/Test Manager – Le Viet Anh – SE02219**: Test Manager is in charge of control test activities and control quality.

- Co-ordinate all testing phases.
- Train team members to use testing tools.
- Control create, review test plan.
- Control execute test phase of project.
- Control quality of project.

***Configuration Manager – Nguyen Xuan Truong - 60465:** Configuration Manager is person who takes responsibilities for configuration management in project.

- Create configuration management plan.

- Ensure that directories in the server repository are correctly structured and correct permissions.
- Train team members to use server when requested via email or during work sessions, team or project manager, project supervisor meetings.
- Perform baseline procedures when requested by the project manager, project supervisor.

***Designer:** Designer role in project team is mainly responsibility for create design document of project.

***Developer:** Developer role in project team is mainly responsibility for create project module, unit test and fix bug of project module.

***Tester:** Tester role in project team is mainly responsibility for create test document of project, execute test phase.

1.3. Tool and Techniques

1.3.1. Tools

Tools				
No	**Name**	**License**	**Description**	**Version**
Integrated Development Environment				
1	Android Development Tools (ADT)	Free	For build Android applications.	ADT 25.6.3
2	Arduino IDE	Free	The open-source Arduino environment makes it easy to write code and upload it to the i/o board. It runs on Windows, Mac OS X, and Linux.	Arduino 1.0.6
Source Control				
3	Gitlap	Free	For source controlling	

Documentation				
4	Microsoft Word	MSDN	For documenting	2013
5	Microsoft Excel	MSDN	For analyzing & statistics	2013
6	Microsoft PowerPoint	MSDN	For presenting	2013
Diagram				
8	Microsoft Visio	MSDN	For UML drawing	2013
9	Skype	Free	For informal communicating	
10	Email (Gmail, Hotmail, FPT University email)	Free	For informal & formal communicating	
Others				
11	Dropbox	Free	For storing resources, documentation For alternative source code backup	
12	Microsoft Project	MSDN	For project management	
Annotation: **MSDN: The license is retrieved from MSDN (Microsoft Developer Network) Free: Free to use** **Trial: Not free to use but can use for free in a period of time** **Premium: Not free to use and need to pay money to use**				

Table 2: Tools

Hardware environment		
No	**Name**	**Description**
1	Laptop	3 laptops with CPU Dual Core i Series, 8 GB RAM with sound card for developing mobile application, running Android Emulator and sound processing.
2	Android Tablet	Android 5.0.1 OS for testing.

3	Android Smart phones	Android 5.0.1 OS for testing.
4	HI-LINK HLK-RM04	This is an embedded module based on the universal serial interface network standard, built-in TCP/IP protocol stack, enabling the user serial port, Ethernet, wireless network (Wi-Fi) interface between the conversions
5	Arduino Pro Mini v3.0	Embedded system development.
6	Basic bell	For testing.
7	Wireless Access Point	Connecting smart home with ASBS system.

Table 3 ***Error! No sequence specified.****: Hardware environment*

1.3.2. Techniques

Hard skills		
No	**Name**	**Description**
1	Java	For tools & mobile application programing
2	XML	For android applications programming
3	C/C++	For Embedded system programming

Table 4: Hard skills

Soft skills	
No	**Name**
1	Presentation skill.
2	Team work skill.
3	Time management skill.
4	Communication skill.
5	Conflict resolve

Table 5: Soft skills

2. Project Management Plan

2.1. Tasks

Software Requirement		
	Hardware (belling system)	Software (mobile application)
Task	Create Software Requirement Specification (SRS) Create master test plan	
Output	SRS	
Deliverable	Deliver SRS , master test plan before 25th June 2015	
Resource Needed	1 people for 2 weeks	

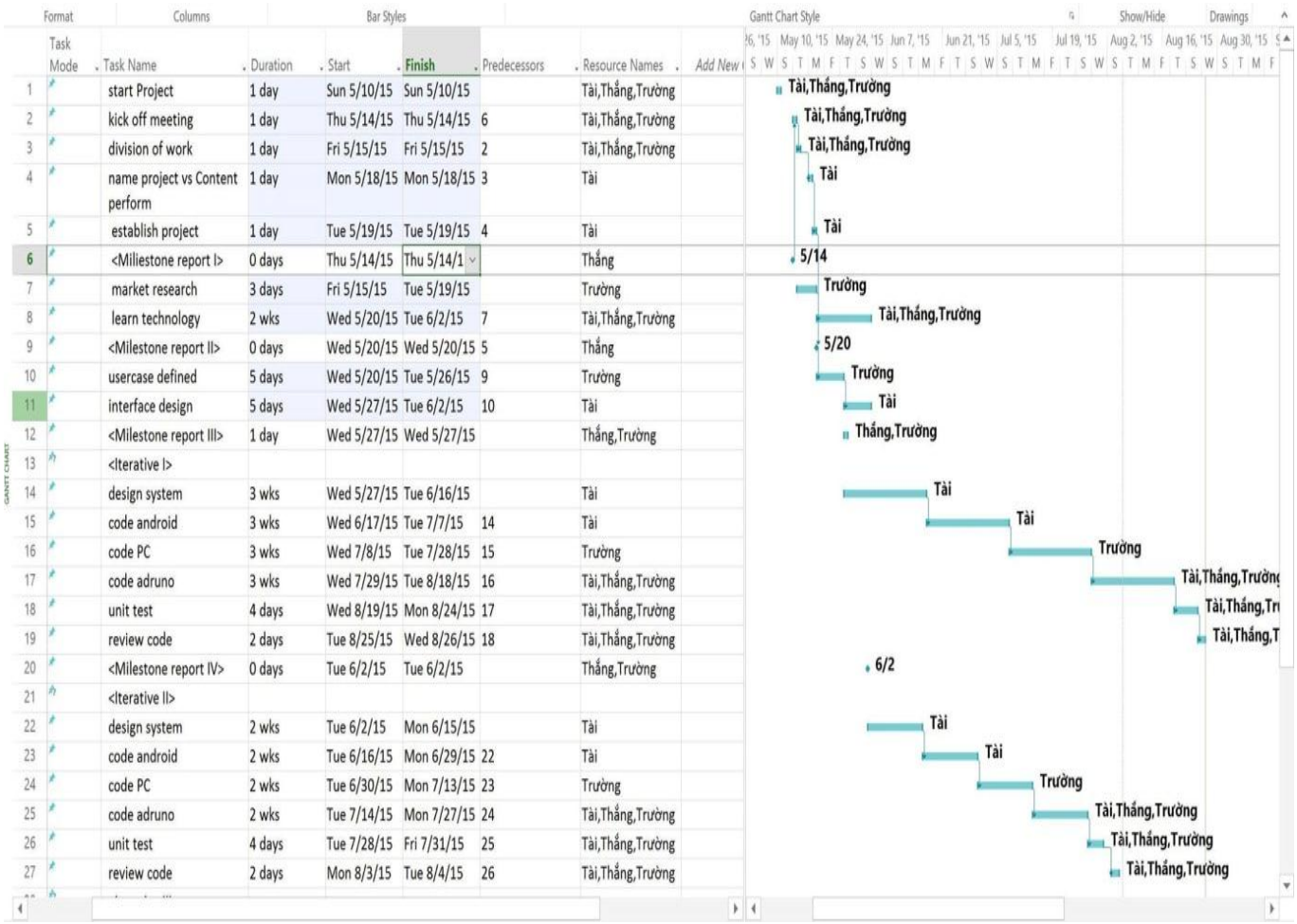
Software Design		
	Hardware (belling system)	Software (mobile application)
Task	Modeling and design architecture for the belling system	Design the system based on OOD paradigm
Output	Architecture Design Document (ADD)	Software design document (SDD)
Deliverable	Deliver ADD before 14th July 2015	Deliver SDD before 14th July 2015
Resource Needed	3 people for 2 weeks	2 people for 2 weeks

Implementation		
	Hardware (belling system)	Software (mobile application)
Task	Coding firmware Create a smart bell	Coding the mobile application
Output	A smart bell.	Mobile application
Deliverable	Deliver before 11th August 2015	Deliver before 11th August 2015
Resource Needed	3 people for 4 weeks, hardware resource	2 people for 4 weeks

Testing		
	Hardware (belling system)	Software (mobile application)
Task	Unit test	Unit test
	System test	

Output	Unit test (for firmware) specification	Unit test specification
	System test specification	
Deliverable	Deliver before 18th August 2014	Deliver before 18st August 2014
Resource Needed	3 people for 1 weeks	2 people for 1 weeks

2.2. Task Sheet: Assignment and timetable



	Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1		start Project	1 day	Sun 5/10/15	Sun 5/10/15		Tài,Thắng,Trường
2		kick off meeting	1 day	Thu 5/14/15	Thu 5/14/15	6	Tài,Thắng,Trường
3		division of work	1 day	Fri 5/15/15	Fri 5/15/15	2	Tài,Thắng,Trường
4		name project vs Content perform	1 day	Mon 5/18/15	Mon 5/18/15	3	Tài
5		establish project	1 day	Tue 5/19/15	Tue 5/19/15	4	Tài
6		<Miliestone report I>	0 days	Thu 5/14/15	Thu 5/14/1		Thắng
7		market research	3 days	Fri 5/15/15	Tue 5/19/15		Trường
8		learn technology	2 wks	Wed 5/20/15	Tue 6/2/15	7	Tài,Thắng,Trường
9		<Milestone report II>	0 days	Wed 5/20/15	Wed 5/20/15	5	Thắng
10		usercase defined	5 days	Wed 5/20/15	Tue 5/26/15	9	Trường
11		interface design	5 days	Wed 5/27/15	Tue 6/2/15	10	Tài
12		<Milestone report III>	1 day	Wed 5/27/15	Wed 5/27/15		Thắng,Trường
13		<Iterative I>					
14		design system	3 wks	Wed 5/27/15	Tue 6/16/15		Tài
15		code android	3 wks	Wed 6/17/15	Tue 7/7/15	14	Tài
16		code PC	3 wks	Wed 7/8/15	Tue 7/28/15	15	Trường
17		code adruno	3 wks	Wed 7/29/15	Tue 8/18/15	16	Tài,Thắng,Trường
18		unit test	4 days	Wed 8/19/15	Mon 8/24/15	17	Tài,Thắng,Trường
19		review code	2 days	Tue 8/25/15	Wed 8/26/15	18	Tài,Thắng,Trường
20		<Milestone report IV>	0 days	Tue 6/2/15	Tue 6/2/15		Thắng,Trường
21		<Iterative II>					
22		design system	2 wks	Tue 6/2/15	Mon 6/15/15		Tài
23		code android	2 wks	Tue 6/16/15	Mon 6/29/15	22	Tài
24		code PC	2 wks	Tue 6/30/15	Mon 7/13/15	23	Trường
25		code adruno	2 wks	Tue 7/14/15	Mon 7/27/15	24	Tài,Thắng,Trường
26		unit test	4 days	Tue 7/28/15	Fri 7/31/15	25	Tài,Thắng,Trường
27		review code	2 days	Mon 8/3/15	Tue 8/4/15	26	Tài,Thắng,Trường

2.3. Coding convention

ASBS Project has been developed using Java, C/C++ programming language. Development team follows Java, C/C++ Coding convention:

- Java Coding convention (Geotechnical Software Service 2011, Java Programming Style Guidelines)
- C/C++ Coding convention (NetBSD 2008a, CCoding standard)(NetBSD 2008b, C++ Coding standard)

- Android Coding convention(Code Style Guidelines for Contributor)

2.4. Dependencies and Constrains

2.3.1. Dependencies

2.3.1.1. Platform Available libraries

Because of constraint on working time and the best approach is to use available libraries on android platform. The functions of the application will be limited as the available libraries.

2.3.1.2. Hardware component

Because ASBS is an embedded project, the quality, the working process of project and the function of systems rely on hardware component. Some hardware component cannot be bought or low quality or too expensive.

2.3.2. Constrains

2.3.2.1. Deadline of project

The project has hard deadline, the team-member will try to bring functions into the project at an early stage. Failure to deliver the inputs could create delays in the project time-line, failure to deliver a standard that meets requirements as expected, and/or removal of requirements from the scope of project

3. Risk

All High level risks must have a mitigation strategy developed and an owner assigned and is monitored on an ongoing basis by the Project Leadership team. Appropriate actions should be taken and ongoing strategy decisions updated as the situation is monitored. All changes must be updated on the Risk Log and final actions should be closed before the project is completed.

All Medium level risks should also have a mitigation strategy developed and an owner assigned; however, review may take place on a less frequent basis. Actions should be monitored and updated and closed before the end of the project. Risks rated Low level do not necessarily need a mitigation strategy to be developed. A regular review need not be carried out but a final assessment must be made before the project can be closed.

ID	Risk Description	Probability (A)	Impact (B)	Rating (A x B)	Mitigation	Owner
1	Estimates are inaccurate	2	2	4	Continues re-estimates as project go on based on personal experience	
2	Change management overload	1	1	1	Focus on what changes are important	
3	Under communication	2	1	2	Working close together to increase communication and reduce wait time	
4	Technology components aren't fit for purpose	2	3	6	Study and use the component as soon as possible to find out if the component is not fit for purpose	
5	Failure to follow methodology	2	2	4	Create clear steps for team-members to follow	
6	Limit of technical knowledge	3	3	9	Increase research time on DSP and music theory, focus on the weak point of team-member knowledge	
7	Member's absence for unknown reasons	2	3	9	Add slack or throat for tasks. Try to keep contact to member daily.	

Probability (A): 0 (very unlikely, < 5%), 1 (unlikely, between 5 and 20%), 2 (likely, 20-60%), 3 (very likely, >60%)

Impact (B): 0 (no expected impact), 1 (minor: budget overrun < 10% and schedule overrun < 10 %), 2 (medium), 3 (major, budget overrun > 30% and/or schedule >30%)

Risk rating (A x B): Level of seriousness. 0 - 2 = Low, 3 - 5 = Medium, 6 - 9 = High

III. System Requirement Specification (SRS)

1. User Requirement Specification

1.1. Control bell system by using mobile application

- Users can turn on/off bell remote using mobile application.
- Users can connect smart phone to modem

- Users can disconnect smart phone from Arduino
- Users can view all bell connect with connected Arduino
- Users can change status for bell.
- Users can change name of bell on application.
- Users can set timer for bell to turn on/turn off (scheduling).
- Users can active/deactivate timer.
- Users can set notification receiver alarm.

1.2. Control bell system by using PC

- Users can turn on/off bell remote using PC.
- Users can connect PC to modem
- Users can disconnect PC from Arduino
- Users can view all bell connect with connected Arduino
- Users can change status for bell.
- Users can change name of bell on application.
- Users can set timer for bell to turn on/turn off (scheduling).
- Users can active/deactivate timer.
- Users can set notification receiver alarm.

2. System Requirement Specification

2.1. External Interface Requirement

2.1.1. User interfaces

User Interface (UI) of mobile application must be design base on Flat UI Design (2014, Wikipedia). In scope of this project, the development team only develops mobile application on Android platform so the UI must design follow Android Design Principle (n.d., Android) and Android Design Patterns (n.d., Android).

2.1.2. Hardware interfaces

The mobile application only runs on smart phone which support Android 4.0+ and Wifi connection.

Arduino (n.d., Arduino) used in the belling system is Arduino Pro Mini 3.0. Module RFXY433 was used for the connection between Arduino and bell.

2.1.3. System interfaces

The mobile connect to Wi-Fi router via Wi-Fi and the rigning system connects to the router via Wi-Fi to create Wireless Local Area Network which allows the mobile to connect to multiple bells in the ringing system.

2.1.4. Communication protocol

Communication between smart phone and UART Wi-Fi Module is via Wi-Fi local using TCP/IP protocol.

Communication between Wi-Fi module and Arduino is serial communications.

Communication Protocol between smart phone and Arduino via special strings called “ASBS command” will be defined in Communication Protocol - ASBS command.

2.2. Functional requirement

2.2.1. General Use Case Diagram

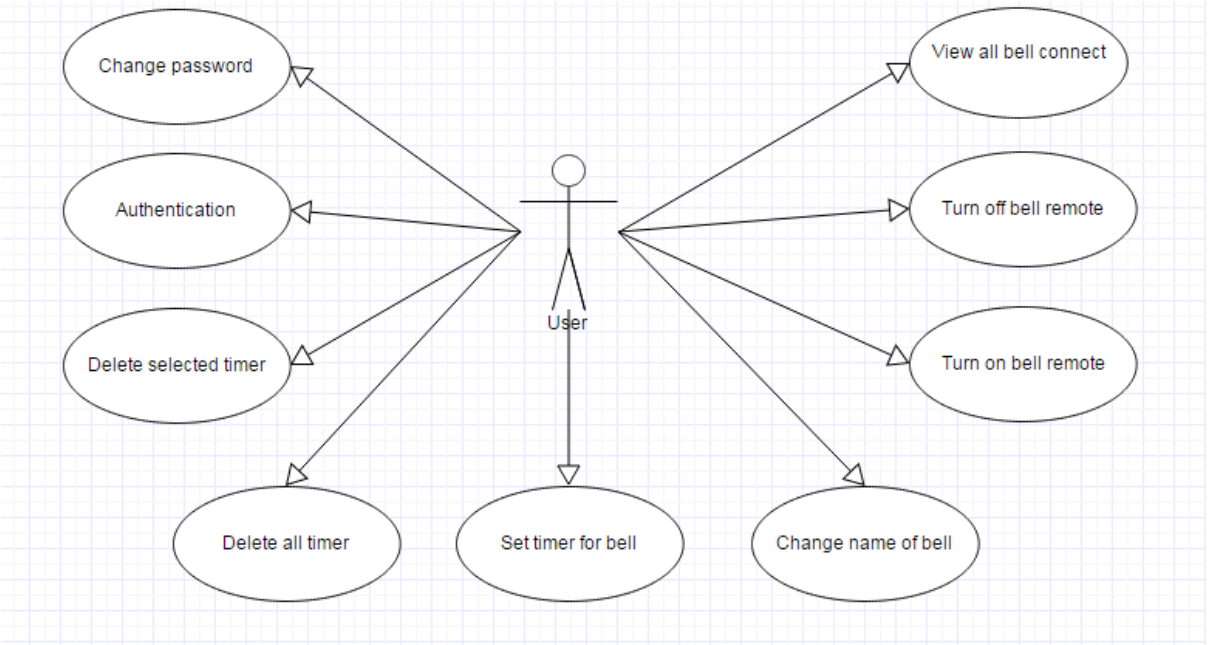


Figure 1:Usercase diagram

2.2.2. Functional Requirement for mobile

All the user’s gesture which interact with smart phone’s touch screen is referenced follow Touch Gesture Reference Guide (2010, Luke Wroblewski).

2.2.2.1.View all bell connect with connected Wi-Fi hub

USE CASE – View all bell connect with connected Wi-Fi hub SPECIFICATION			
Use-case No.	1.2.1	**Use-case Version**	1.0
Use-case Name	View all bell connect with connected Wi-Fi hub		

Author	Pham Nam Thang		
Date	27-06-2015	**Priority**	High
Related Prototype			

Actor:
User
Description:
Allow the user view all bell connect with connected Wi-Fi hub. Each bell item have four status:

- Stop ringing: A red button with blue border.
- Scheduled ring: A yellow button with blue border.
- Ringing: A green button with blink blue border.
- Disable ring: A gray button with blue border.

Trigger:
Pres-conditions:
- The user start mobile application.
- The user's mobile phone connects to Wi-Fi hub successfully.

Post-conditions:
None
Main Success Scenario:

Step	Actor action	System respond
1	The user taps on mobile application icon.	1. loading_screen appear. 2. Message "Connect to Arduino successfully" appear. 3. List of bells connect to connected Wi-Fi hub appear.

Alternative Scenario No. n:
None
Exceptions:

Exceptions	System respond
Can not connect to Wifi hub	1. Show message "Cannot connect to Arduino" with "reconnect" button. 2. When user click on "reconnect" button, reconnect to wifi hub.

Notes and Issues:

2.2.2.2.Turn on bell remote using mobile application

USE CASE – Turn on bell remote using mobile application SPECIFICATION			
Use-case No.	1.2.2	**Use-case Version**	1.0

Use-case Name	Turn on bell remote using mobile application		
Author	Pham Nam Thang		
Date	27-06-2015	**Priority**	High
Related Prototype	Main_screen		

Actor:

User

Description:

Allow the user to turn on bell using mobile application.

Trigger:

Pres-conditions:

Main success scenario UC No.1.2.1

Post-conditions:

The bell is turned on.

Main Success Scenario:

Step	Actor action	System respond
1	User tap on ON/OFF button (OFF state) at the row represent to the bell at main_screen.	Show message “Do you want to turn on ringing”. Show “turn on” button.
2	User click on “turn on” button.	Button ON/OFF at main_screen transfer to “ON” state. The bell is turned on.

Alternative Scenario No. n:

None

Exceptions:

Exceptions	System respond
User tap on ON/OFF button (OFF state) at the row represent to the bell at main_screen but the build was turned off manually.	1. Show message “This bell was turned off manually” 2. The button ON/OFF at main screen is still at “OFF” state.
User tap on ON/OFF button (OFF state) at the row represent to the bell at main_screen but the build was disable.	1. Show message “This bell was disabled” 2. The button ON/OFF at main screen is still at “Disable” state.

Notes and Issues:

2.2.2.2.Turn off bell remote using mobile application

USE CASE – Turn off bell remote using mobile application SPECIFICATION

Use-case No.	1.2.2	Use-case Version	1.0
Use-case Name	Turn off bell remote using mobile application		
Author	Pham Nam Thang		
Date	27-06-2015	Priority	High
Related Prototype	Main_screen		

Actor:

User

Description:

Allow the user to turn off bell using mobile application.

Trigger:

Pres-conditions:

1. Main success scenario UC No. 1.2.1
2. The bell is turned on.

Post-conditions:

The bell is turned off.

Main Success Scenario:

Step	Actor action	System respond
1	User tap on ON/OFF button (ON state) at the row represent to the bell at main_screen.	Show message “Do you want to turn off ringing”. Show “turn off” button.
2	User click on “turn off” button.	Button ON/OFF at main_screen transfer to “OFF” state. The bell is turned off.

Alternative Scenario No. n:

None

Exceptions:

Exceptions	System respond
User tap on ON/OFF button (ON state) at the row represent to the bell at main_screen but the build was turned off manually.	1. Show message “This bell was turned on manually” 2. The button ON/OFF at main screen is still at “ON” state.
User tap on ON/OFF button (OFF state) at the row represent to the bell at main_screen but the build was disable.	1. Show message “This bell was disabled” 2. The button ON/OFF at main screen is still at “Disable” state.

Notes and Issues:

2.2.2.4.Change name of bell on application

USE CASE – Change name of bell on application SPECIFICATION

Use-case No.	1.2.4	**Use-case Version**	1.0
Use-case Name	Change name of bell on application		
Author	Pham Nam Thang		
Date	27-06-2015	**Priority**	High
Related Prototype			

Actor:

User

Description:

Allow the user change the name represented for the bell. The bell name include text character and number.

- This name have not over five character.
- This name have to start with text character.
- This name allow only alphabet and numberic character.
- Don't have any two bell have the same name.

Trigger:

Pres-conditions:

UC No. 1.2.3 Main Success Scenario

Post-conditions:

The name of selected bells is changed.

Main Success Scenario:

Step	Actor action	System respond
1	The user presses (taps and hold)on bell.	Edit name dialog appears.
2	The user tap on text field in edit name dialog.	Keyboard appears.
3	The user input new name for bell and tap "OK" button in edit name dialog.	Edit name dialog appears. The name represent for the bell in main_screen is changed. Soft keyboard disappears.

Alternative Scenario No.1:

Step	Actor action	System respond
1	The user presses on bell.	Edit name dialog appear.
2	The user tap on text field in edit name dialog.	Keyboard appears.
3	The user input new name for bell and tap "Cancel" button.	Edit name dialog appear. The name represent for the bell in main_screen is not changed. Soft keyboard disappears.

Alternative Scenario No.1:

Step	Actor action	System respond
1	The user presses on bell.	Edit name dialog appear.
2	The user tap on text field in edit name dialog.	Keyboard appears.
3	The user input new name for bell and tap "OK" button in edit name dialog. The new name is the same name with previous name of this bell.	Edit name dialog appear. The name represent for the bell in main_screen is not changed. Soft keyboard disappears.

Exceptions:

Exceptions	System respond
User input much more than eight character.	Do not allow user input more than five character.
User input non alphabet character or non numberic character.	After user click on "OK" button in edit name dialog, show message "Please input only alphabet character or numberic character"
User clear text in edit text and click on "OK" button in edit name dialog.	Show message "Do not using empty name"
User input the same name with another bell.	Show message "This name is already in use by another bell"
Two user input the same name with a bell or another bell by two diffence mobile device.	The slower user device show message "This name is already in use by another bell"
User press on bell and slide to another screen during the time of the edit text dialog showing.	Disappears the edit name dialog.

Notes and Issues:

2.2.2.5.Set timer for bell

USE CASE – Set timer for bell SPECIFICATION			
Use-case No.	1.2.5	**Use-case Version**	1.0
Use-case Name	Set timer for bell		
Author	Pham Nam Thang		
Date	27-06-2015	**Priority**	High

Related Prototype	

Actor:
User
Description:
Allow the user set timer (action and the time for action) for bell. User will be created ringing schedule.
Trigger:
Pres-conditions:
UC No. 1.2.1 Main Success Scenario
Post-conditions:
Timer is set for selected bells.
Main Success Scenario:

Step	Actor action	System respond
1	The user slide to “schedule” screen.	Show schedule screen with schedule list.
2	The user taps on “add” button in the Timer option bar.	timer_bell_screenappears.
3	The user taps on “Start time” time picker to adjust time.	
4	The user choose in group of check box to sellect bell in this schedule.	
5	The user check on “Turn ON” check box to select the mode for timer.	
6	The user taps on “MON” or “TUE” or “WED”…..or “SUN” button	Tapped button’s color is changed.
7	The user taps on “During” time picker to adjust time.	
8	The user click on time schedule to input repeat option.	Show list option: - Repeat forever. - Repeat n times. - No repeat.
9	User choose repeat forever.	Repeat times input box disappear.
10	The user tap on “OK” in timer_bell_screen.	Main_screen appears with input selected time, repeat day and icon represented to

		timer status (mode, changing mode) is showed at selected bell's Timer option bar appears (below the selected bell). Active/De-active button is showed at De-active status.

Alternative Scenario No. 1:

Step	Actor action	System respond
1	The user slide to "schedule" screen.	Show schedule screen with schedule list.
2	The user taps on "add" button in the Timer option bar.	timer_bell_screenappears.
3	The user taps on "Start time" time picker to adjust time.	
4	The user choose in group of check box to sellect bell in this schedule.	
5	The user check on "Turn ON" check box to select the mode for timer.	
6	The user taps on "MON" or "TUE" or "WED".....or "SUN" button	Tapped button's color is changed.
7	The user taps on "During" time picker to adjust time.	
8	The user click on time schedule to input repeat option.	Show list option: - Repeat forever. - Repeat n times. - No repeat.
9	User choose repeat n times.	Repeat times input box appear.
10	The user tap on text field in edit repeat times.	Keyboard appears.
11	The user input repeat times.	
12	The user tap on "OK" in timer_bell_screen.	Main_screen appears with input selected time, repeat day and icon represented to timer status (mode, changing mode) is

		showed at selected bell's Timer option bar appears (below the selected bell). Active/De-active button is showed at De-active status.

Exceptions:

Exceptions	System respond

Notes and Issues:

2.2.2.6.Delete all timer

USE CASE – Delete all timer SPECIFICATION			
Use-case No.	1.2.6	**Use-case Version**	1.0
Use-case Name	Delete all timer		
Author	Pham Nam Thang		
Date	27-06-2015	**Priority**	High
Related Prototype			

Actor:
User
Description:
Allow the user delete all timer in schedule screen.
Trigger:
Pres-conditions:
1. UC No. 1.2.1 Main Success Scenario
2. UC No. 1.2.5 Main Success Scenario
Post-conditions:
Clear schedule screen.
Main Success Scenario:

Step	Actor action	System respond
1	The user click on "Delete all" button.	Show dialog "Do you want to clear all list schedule?"
2	The user tap on "OK" button.	Clear all timer on the screen.

Alternative Scenario No. n:
None
Exceptions:

Exceptions	System respond

Notes and Issues:	

2.2.2.7.Delete selected timer

USE CASE – Delete selected timer SPECIFICATION			
Use-case No.	1.2.7	**Use-case Version**	1.0
Use-case Name	Delete selected timer		
Author	Pham Nam Thang		
Date	27-06-2015	**Priority**	High
Related Prototype			

Actor:
User
Description:
Allow the user delete all timer in schedule screen.
Trigger:
Pres-conditions:
1. UC No. 1.2.1 Main Success Scenario
2. UC No. 1.2.5 Main Success Scenario
Post-conditions:
Remove selected timer from schedule screen.
Main Success Scenario:

Step	Actor action	System respond
1	The user click on “Edit” button.	Show check box to the left of timer item.
2	The user check on “delete selected” check box.	Show dialog “Do you want to delete this timer?”
3	The user tap on “OK” button.	Remove selected timer from list.

Alternative Scenario No. 1:

Step	Actor action	System respond
1	The user click on “Edit” button.	Show check box to the left of timer item.
2	The user check on “delete selected” check box of the last timer.	Show dialog “Do you want to delete this timer?”
3	The user tap on “OK” button.	Clear all timer on the screen.

Exceptions:

Exceptions	System respond

Notes and Issues:

2.2.2.8. Authentication

USE CASE – Authentication SPECIFICATION			
Use-case No.	1.2.8	**Use-case Version**	1.0
Use-case Name	Authenticaton		
Author	Pham Nam Thang		
Date	27-06-2015	**Priority**	High
Related Prototype			

Actor:

User

Description:

Allow user send password to authentication. Allow the user input a password to authentication:

- This password have not over twelve character.
- This password have to have atlease six character.
- This name allow only alphabet and numberic character.

Trigger:

Pre-conditions 1:

1. UC No. 1.2.1 Main Success Scenario.

Post-conditions:

User could use all of another feature.

Main Success Scenario:

Step	Actor action	System respond
1	User tap on edit text password.	Keyboard appears.
2	User input password.	Show bullet symbol.
3	User click on “Submit” button.	Show “Authentication success” dialog.

Alternative Scenario No. n:

None

Exceptions:

Exceptions	System respond

User input much more than eight character.	Do not allow user input more than twelve character.
User input not enough six character.	Show message “Wrong password format”.
User input non alphabet character or non numberic character.	After user click on “OK”, show message “Wrong password format”.
Notes and Issues:	

2.2.2.9.Change password

USE CASE – Change password SPECIFICATION			
Use-case No.	1.2.9	**Use-case Version**	1.0
Use-case Name	Change password		
Author	Pham Nam Thang		
Date	27-06-2015	**Priority**	High
Related Prototype			

Actor:
User
Description:
Allow user send password to authentication. Allow the user input a password to authentication:

- This password have not over twelve character.
- This password have to have atlease six character.
- This name allow only alphabet and numberic character.

Trigger:
Pre-conditions 1:
1. UC No. 1.2.1 Main Success Scenario.
2. UC No. 1.2.8 Main Success Scenario.

Post-conditions:
User could change password.
Main Success Scenario:

Step	Actor action	System respond
1	User click on “Change password” button.	The edit text “change password” appear. Button “OK” appear. Button “Cancel” appear.
2	User tap on edit text password.	Keyboard appears.
3	User input password.	Show bullet symbol.
4	User click on “OK” button.	Show “Change password success” dialog.

Alternative Scenario No. n:

Step	Actor action	System respond
1	User click on “Change password” button.	The edit text “change password” appear. Button “OK” appear. Button “Cancel” appear.
2	User tap on edit text password.	Keyboard appears.
3	User input password.	Show bullet symbol.
4	User click on “Cancel” button.	The edit text “change password” disappear. Button “OK” disappear. Button “Cancel” disappear.

Exceptions:

Exceptions	System respond
User input much more than eight character.	Do not allow user input more than twelve character.
User input not enough six character.	Show message “Wrong password format”.
User input non alphabet character or non numberic character.	After user click on “OK”, show message “Wrong password format”.
User input the same password with previous password.	Don’t do anything.

Notes and Issues:

2.2.2.10.Message list

Message
Connect to Arduino successfully.
Cannot connect to Arduino.
Do you want to turn on ringing.
This bell was turned off manually.
Do you want to turn off ringing.
This bell was turned on manually.
This bell was disabled.
Please input only alphabet character or numberic character.
Do not using empty name.
This name is already in use by another bell.
Do you want to clear all list schedule?
Do you want to delete this timer?
Authentication success.
Wrong password format.
Change password success.

2.2.3. Functional Requirement for desktop

2.2.3.1.View all bell connect with connected Wi-Fi hub

USE CASE – View all bell connect with connected Wi-Fi hub SPECIFICATION			
Use-case No.	1.3.1	**Use-case Version**	1.0
Use-case Name	View all bell connect with connected Wi-Fi hub		
Author	Pham Nam Thang		
Date	27-06-2015	**Priority**	High
Related Prototype			

Actor:

User

Description:

Allow the user view all bell connect with connected Wi-Fi hub. Each bell item have four status:

- Stop ringing: A red button with blue border.
- Scheduled ring: A yellow button with blue border.
- Ringing: A green button with blink blue border.
- Disable ring: A gray button with blue border.

Trigger:

Pres-conditions:

- The user start mobile application.
- The user's mobile phone connects to Wi-Fi hub successfully.

Post-conditions:

None

Main Success Scenario:

Step	Actor action	System respond
1	The user taps on mobile application icon.	1. loading_screen appear. 2. Message "Connect to Arduino successfully" appear. 3. List of bells connect to connected Wi-Fi hub appear.

Alternative Scenario No. n:

None

Exceptions:

Exceptions	System respond
Can not connect to Wifi hub	1. Show message "Cannot connect to Arduino" with "reconnect" button. 2. When user click on "reconnect" button,

	reconnect to wifi hub.
Notes and Issues:	

2.2.3.2.Turn on bell remote using mobile application

USE CASE – Turn on bell remote using mobile application SPECIFICATION			
Use-case No.	1.3.2	**Use-case Version**	1.0
Use-case Name	Turn on bell remote using mobile application		
Author	Pham Nam Thang		
Date	27-06-2015	**Priority**	High
Related Prototype	Main_screen		

Actor:
User
Description:
Allow the user to turn on bell using mobile application.
Trigger:
Pres-conditions:
Main success scenario UC No.1.3.1
Post-conditions:
The bell is turned on.
Main Success Scenario:

Step	Actor action	System respond
1	User tap on ON/OFF button (OFF state) at the row represent to the bell at main_screen.	Show message “Do you want to turn on ringing”. Show “turn on” button.
2	User click on “turn on” button.	Button ON/OFF at main_screen transfer to “ON” state. The bell is turned on.

Alternative Scenario No. n:
None
Exceptions:

Exceptions	System respond
User tap on ON/OFF button (OFF state) at the row represent to the bell at main_screen but the build was turned off manually.	1. Show message “This bell was turned off manually” 2. The button ON/OFF at main screen is still at “OFF” state.
User tap on ON/OFF button (OFF state) at	1. Show message “This bell was disabled”

the row represent to the bell at main_screen but the build was disable.	2. The button ON/OFF at main screen is still at “Disable” state.
Notes and Issues:	

2.2.3.2.Turn off bell remote using mobile application

USE CASE – Turn off bell remote using mobile application SPECIFICATION			
Use-case No.	1.3.2	**Use-case Version**	1.0
Use-case Name	Turn off bell remote using mobile application		
Author	Pham Nam Thang		
Date	27-06-2015	**Priority**	High
Related Prototype	Main_screen		

Actor:
User
Description:
Allow the user to turn off bell using mobile application.
Trigger:
Pres-conditions:
1. Main success scenario UC No. 1.3.1
2. The bell is turned on.
Post-conditions:
The bell is turned off.
Main Success Scenario:

Step	Actor action	System respond
1	User tap on ON/OFF button (ON state) at the row represent to the bell at main_screen.	Show message “Do you want to turn off ringing”. Show “turn off” button.
2	User click on “turn off” button.	Button ON/OFF at main_screen transfer to “OFF” state. The bell is turned off.

Alternative Scenario No. n:
None
Exceptions:

Exceptions	System respond
User tap on ON/OFF button (ON state) at the row represent to the bell at main_screen but the build was turned off manually.	1. Show message “This bell was turned on manually” 2. The button ON/OFF at main screen is still at “ON” state.

User tap on ON/OFF button (OFF state) at the row represent to the bell at main_screen but the build was disable.	1. Show message “This bell was disabled” 2. The button ON/OFF at main screen is still at “Disable” state.
Notes and Issues:	

2.2.3.4.Change name of bell on application

USE CASE – Change name of bell on application SPECIFICATION			
Use-case No.	1.3.4	**Use-case Version**	1.0
Use-case Name	Change name of bell on application		
Author	Pham Nam Thang		
Date	27-06-2015	**Priority**	High
Related Prototype			

Actor:
User
Description:
Allow the user change the name represented for the bell. The bell name include text character and number.

- This name have not over five character.
- This name have to start with text character.
- This name allow only alphabet and numberic character.
- Don’t have any two bell have the same name.

Trigger:
Pres-conditions:
UC No. 1.3.3 Main Success Scenario
Post-conditions:
The name of selected bells is changed.
Main Success Scenario:

Step	Actor action	System respond
1	The user presses (taps and hold)on bell.	Edit name dialog appears.
2	The user tap on text field in edit name dialog.	Keyboard appears.
3	The user input new name for bell and tap “OK” button in edit name dialog.	Edit name dialog appears. The name represent for the bell in main_screen is changed. Soft keyboard disappears.

Alternative Scenario No.1:

Step	Actor action	System respond
1	The user presses on bell.	Edit name dialog appear.
2	The user tap on text field in edit name dialog.	Keyboard appears.
3	The user input new name for bell and tap “Cancel” button.	Edit name dialog appear. The name represent for the bell in main_screen is not changed. Soft keyboard disappears.

Alternative Scenario No.1:

Step	Actor action	System respond
1	The user presses on bell.	Edit name dialog appear.
2	The user tap on text field in edit name dialog.	Keyboard appears.
3	The user input new name for bell and tap “OK” button in edit name dialog. The new name is the same name with previous name of this bell.	Edit name dialog appear. The name represent for the bell in main_screen is not changed. Soft keyboard disappears.

Exceptions:

Exceptions	System respond
User input much more than eight character.	Do not allow user input more than five character.
User input non alphabet character or non numberic character.	After user click on “OK” button in edit name dialog, show message “Please input only alphabet character or numberic character”
User clear text in edit text and click on “OK” button in edit name dialog.	Show message “Do not using empty name”
User input the same name with another bell.	Show message “This name is already in use by another bell”
Two user input the same name with a bell or another bell by two diffence mobile device.	The slower user device show message “This name is already in use by another bell”
User press on bell and slide to another screen during the time of the edit text dialog showing.	Disappears the edit name dialog.

Notes and Issues:

2.2.3.5.Set timer for bell

USE CASE – Set timer for bell SPECIFICATION			
Use-case No.	1.3.5	**Use-case Version**	1.0
Use-case Name	Set timer for bell		
Author	Pham Nam Thang		
Date	27-06-2015	**Priority**	High
Related Prototype			

Actor:
User
Description:
Allow the user set timer (action and the time for action) for bell. User will be created ringing schedule.
Trigger:
Pres-conditions:
UC No. 1.2.1 Main Success Scenario
Post-conditions:
Timer is set for selected bells.
Main Success Scenario:

Step	Actor action	System respond
1	The user slide to “schedule” screen.	Show schedule screen with schedule list.
2	The user taps on “add” button in the Timer option bar.	timer_bell_screenappears.
3	The user taps on “Start time” time picker to adjust time.	
4	The user choose in group of check box to sellect bell in this schedule.	
5	The user check on “Turn ON” check box to select the mode for timer.	
6	The user taps on “MON” or “TUE” or “WED”…..or “SUN” button	Tapped button’s color is changed.
7	The user taps on “During” time picker to adjust time.	

8	The user click on time schedule to input repeat option.	Show list option: - Repeat forever. - Repeat n times. - No repeat.
9	User choose repeat forever.	Repeat times input box disappear.
10	The user tap on “OK” in timer_bell_screen.	Main_screen appears with input selected time, repeat day and icon represented to timer status (mode, changing mode) is showed at selected bell’s Timer option bar appears (below the selected bell). Active/De-active button is showed at De-active status.

Alternative Scenario No. 1:

Step	Actor action	System respond
1	The user slide to “schedule” screen.	Show schedule screen with schedule list.
2	The user taps on “add” button in the Timer option bar.	timer_bell_screenappears.
3	The user taps on “Start time” time picker to adjust time.	
4	The user choose in group of check box to sellect bell in this schedule.	
5	The user check on “Turn ON” check box to select the mode for timer.	
6	The user taps on “MON” or “TUE” or “WED”…..or “SUN” button	Tapped button’s color is changed.
7	The user taps on “During” time picker to adjust time.	
8	The user click on time schedule to input repeat option.	Show list option: - Repeat forever. - Repeat n times. - No repeat.
9	User choose repeat n	Repeat times input box appear.

	times.	
10	The user tap on text field in edit repeat times.	Keyboard appears.
11	The user input repeat times.	
12	The user tap on “OK” in timer_bell_screen.	Main_screen appears with input selected time, repeat day and icon represented to timer status (mode, changing mode) is showed at selected bell’s Timer option bar appears (below the selected bell). Active/De-active button is showed at De-active status.

Exceptions:

Exceptions	System respond

Notes and Issues:

2.2.3.6.Delete all timer

USE CASE – Delete all timer SPECIFICATION			
Use-case No.	1.3.6	**Use-case Version**	1.0
Use-case Name	Delete all timer		
Author	Pham Nam Thang		
Date	27-06-2015	**Priority**	High
Related Prototype			

Actor:
User
Description:
Allow the user delete all timer in schedule screen.
Trigger:
Pres-conditions:
1. UC No. 1.2.1 Main Success Scenario
2. UC No. 1.2.5 Main Success Scenario
Post-conditions:
Clear schedule screen.
Main Success Scenario:

Step	Actor action	System respond
1	The user click on “Delete	Show dialog “Do you want to clear all list

	all" button.	schedule?"
2	The user tap on "OK" button.	Clear all timer on the screen.

Alternative Scenario No. n:
None
Exceptions:

Exceptions	System respond

Notes and Issues:

2.2.3.7.Delete selected timer

USE CASE – Delete selected timer SPECIFICATION			
Use-case No.	1.3.7	**Use-case Version**	1.0
Use-case Name	Delete selected timer		
Author	Pham Nam Thang		
Date	27-06-2015	**Priority**	High
Related Prototype			

Actor:
User
Description:
Allow the user delete all timer in schedule screen.
Trigger:
Pres-conditions:
1. UC No. 1.2.1 Main Success Scenario
2. UC No. 1.2.5 Main Success Scenario
Post-conditions:
Remove selected timer from schedule screen.
Main Success Scenario:

Step	Actor action	System respond
1	The user click on "Edit" button.	Show check box to the left of timer item.
2	The user check on "delete selected" check box.	Show dialog "Do you want to delete this timer?"
3	The user tap on "OK" button.	Remove selected timer from list.

Alternative Scenario No. 1:

Step	Actor action	System respond
1	The user click on “Edit” button.	Show check box to the left of timer item.
2	The user check on “delete selected” check box of the last timer.	Show dialog “Do you want to delete this timer?”
3	The user tap on “OK” button.	Clear all timer on the screen.

Exceptions:

Exceptions	System respond

Notes and Issues:

2.2.3.8. Authentication

USE CASE – Authentication SPECIFICATION			
Use-case No.	1.3.8	**Use-case Version**	1.0
Use-case Name	Authenticaton		
Author	Pham Nam Thang		
Date	27-06-2015	**Priority**	High
Related Prototype			

Actor:
User
Description:
Allow user send password to authentication. Allow the user input a password to authentication:

- This password have not over twelve character.
- This password have to have atlease six character.
- This name allow only alphabet and numberic character.

Trigger:
Pre-conditions 1:
1. UC No. 1.2.1 Main Success Scenario.
Post-conditions:
User could use all of another feature.
Main Success Scenario:

Step	Actor action	System respond
1	User tap on edit text password.	Keyboard appears.

2	User input password.	Show bullet symbol.
3	User click on “Submit” button.	Show “Authentication success” dialog.

Alternative Scenario No. n:
None
Exceptions:

Exceptions	System respond
User input much more than eight character.	Do not allow user input more than twelve character.
User input not enough six character.	Show message “Wrong password format”.
User input non alphabet character or non numberic character.	After user click on “OK”, show message “Wrong password format”.

Notes and Issues:

2.2.3.9.Change password

USE CASE – Change password SPECIFICATION			
Use-case No.	1.3.9	**Use-case Version**	1.0
Use-case Name	Change password		
Author	Pham Nam Thang		
Date	27-06-2015	**Priority**	High
Related Prototype			

Actor:
User
Description:
Allow user send password to authentication. Allow the user input a password to authentication:

- This password have not over twelve character.
- This password have to have atlease six character.
- This name allow only alphabet and numberic character.

Trigger:
Pre-conditions 1:
1. UC No. 1.2.1 Main Success Scenario.
2. UC No. 1.2.8 Main Success Scenario.
Post-conditions:
User could change password.
Main Success Scenario:

Step	Actor action	System respond

1	User click on “Change password” button.	The edit text “change password” appear. Button “OK” appear. Button “Cancel” appear.
2	User tap on edit text password.	Keyboard appears.
3	User input password.	Show bullet symbol.
4	User click on “OK” button.	Show “Change password success” dialog.

Alternative Scenario No. n:

Step	Actor action	System respond
1	User click on “Change password” button.	The edit text “change password” appear. Button “OK” appear. Button “Cancel” appear.
2	User tap on edit text password.	Keyboard appears.
3	User input password.	Show bullet symbol.
4	User click on “Cancel” button.	The edit text “change password” disappear. Button “OK” disappear. Button “Cancel” disappear.

Exceptions:

Exceptions	System respond
User input much more than eight character.	Do not allow user input more than twelve character.
User input not enough six character.	Show message “Wrong password format”.
User input non alphabet character or non numberic character.	After user click on “OK”, show message “Wrong password format”.
User input the same password with previous password.	Don’t do anything.

Notes and Issues:

2.2.3.10.Message list

Message
Connect to Arduino successfully.
Cannot connect to Arduino.
Do you want to turn on ringing.
This bell was turned off manually.
Do you want to turn off ringing.
This bell was turned on manually.

This bell was disabled.
Please input only alphabet character or numberic character.
Do not using empty name.
This name is already in use by another bell.
Do you want to clear all list schedule?
Do you want to delete this timer?
Authentication success.
Wrong password format.
Change password success.

2.3. Non-functional Requirement

2.3.1. Reliability

- Status of group bell must be the same as status displayed on the mobile application.
- Mobile application must work correctly with belling system, have no conflict between devices.
- Belling system must work constantly under high temperature and humidity, especially in Vietnam with monsoon (Embassy of Vietnam, 2007).

2.3.2. Availability

- The mobile application is easy to connect to belling system to transform data.
- All group bell in connected belling system must be operated.
- Power of components like include Wireless Access Point, Master System, Bell, and Smart Phone is stable and available.

2.3.3. Security

- Because of using Wi-Fi connection, the connection between the smart phone and the belling system is protected by Wi-Fi security protocols (Wikipedia, 2014).

2.3.4. Maintainability

- Implementation code must be follow coding standard, clearly commented for maintaining and enhancing system (in the future).
- All group bell in belling system can be replaced easily.

2.3.5. Portability

- The mobile application can be installed on a smart phone or tablet.

2.3.6. Performance

- 60 second is maximum time for mobile application to connect to belling system and operate available group bell in the belling system
- System must response a request from user in maximum 2 seconds.

2.4. Prototype

2.4.1. Belling list screen

This screen appears when user open the application. This screen have a sliding menu bar, a list of belling button with four status:

- Stoping
- Preparing
- Ringing
- Disable

Each button have to showing 3 information include button name, next ringing schedule and status of button.

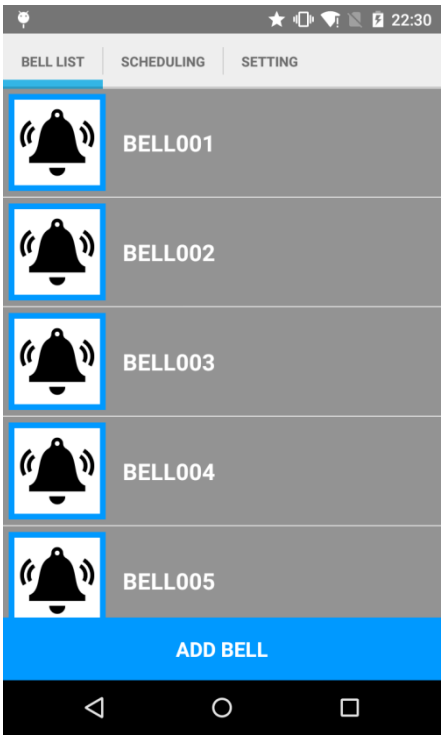


Figure 2: Belling list screen

2.4.2. Change to ringing

When user click on a stoping button or prepareing button, this button will be change their status to ringing. After that, the application sending a ringing message to belling system.

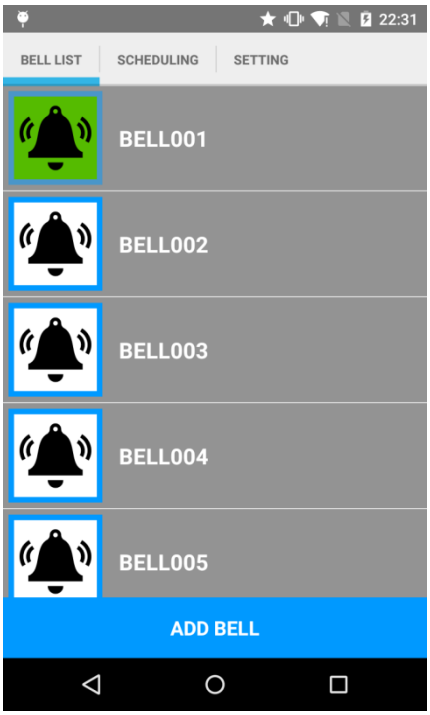


Figure 3:Change to ringing

2.4.3. Change to stop ringing

When user click on a ringing button, this button will be change their status to stoping. After that, the application sending a stop ringing message to belling system.

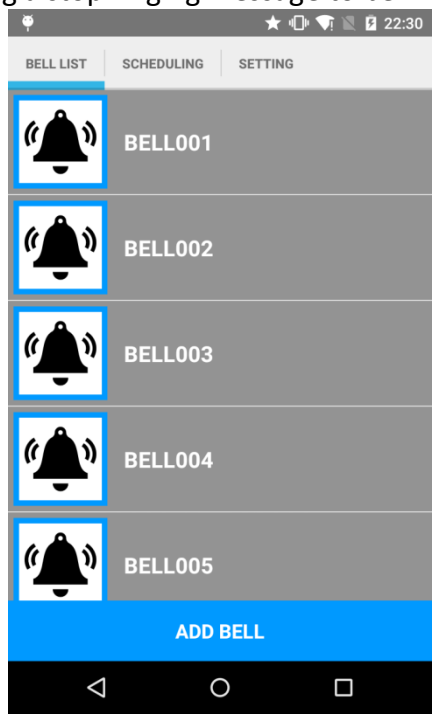


Figure 4:Change to stop ringing

2.4.4. Change belling dialog

When user long clicked on bell name. A change name dialog will apppear.

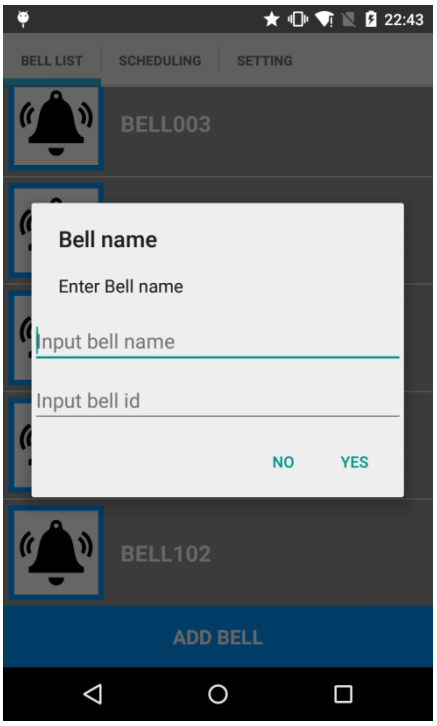


Figure 5:Change bell name dialog

2.4.5. Input belling name

User could be input new bell name.

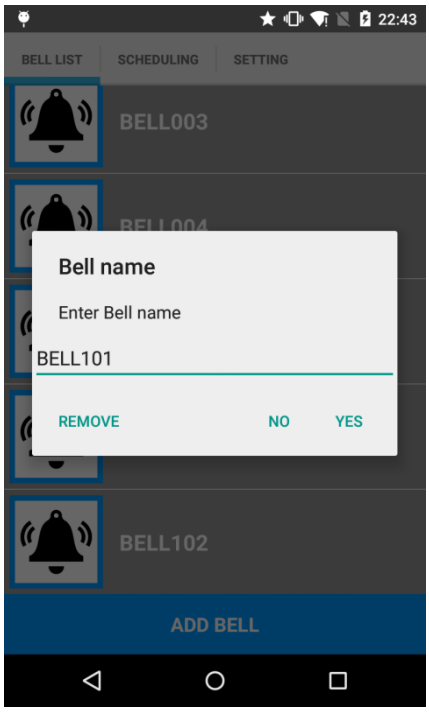


Figure 6:Input belling name

2.4.6. Change bell name success

When user clicked on ok button, the disalog will disappear. The button change their name and a new name message will send to bell system. After changing bell name success. The bell button have a new name instead.

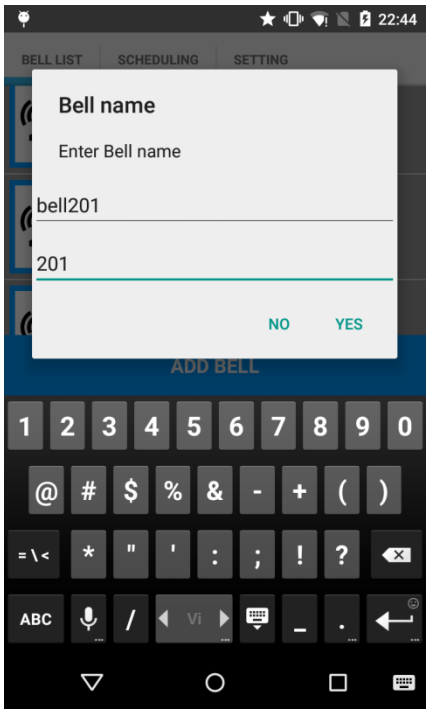


Figure 7:Chage bell name success.

2.4.7. Long name error message

If user choose a long name and input more than 5 character, a message will be show.

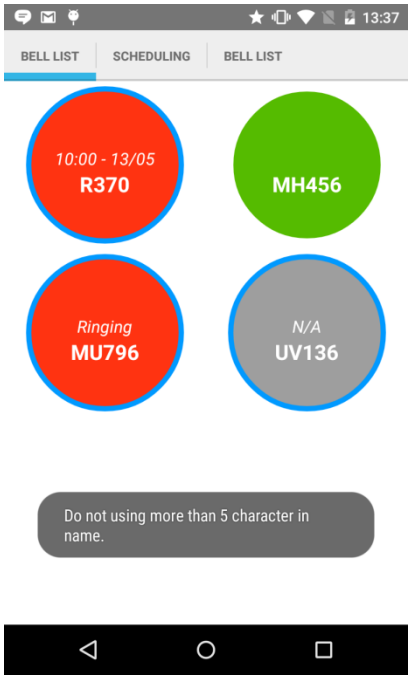


Figure 8:Long name error message

2.4.8. Contain space character error

If user input non alphabet and numberic character, a message will be show.

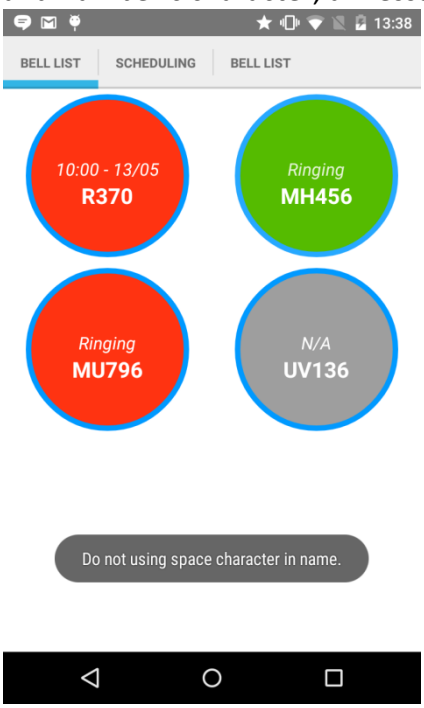


Figure 9:Contain space character error.

2.4.9. Schedule list screen

When user sliding screen to right, scheduling screen will be showing. It have a control bar include three button:

- Add: Add new schedule item
- Delete all: Delete all current schedule item
- Edit: Choose button to delete the item

Each scheduling item showing:

- Start time: Time to start ringing. Format hh:mm:AM
- Repeat times: Times to repeat the scheduling.
- Day in week: List of day in week

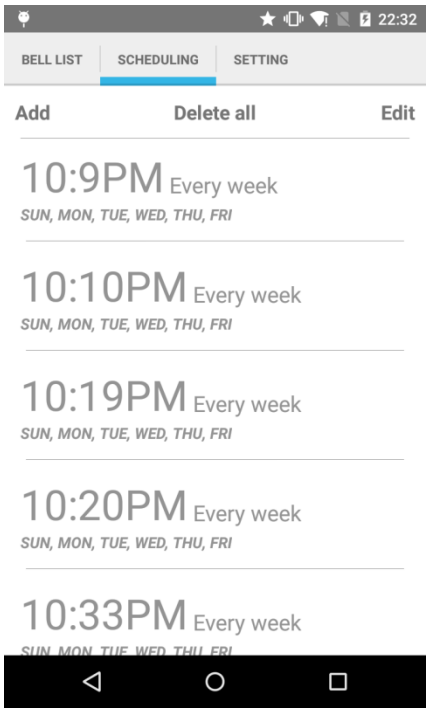


Figure 10:Schedule list screen.

2.4.10. Edit scheduling item

When user clicked on “Edit” button, it change to “Done” button and each scheduling item have an “On/Off” switcher.

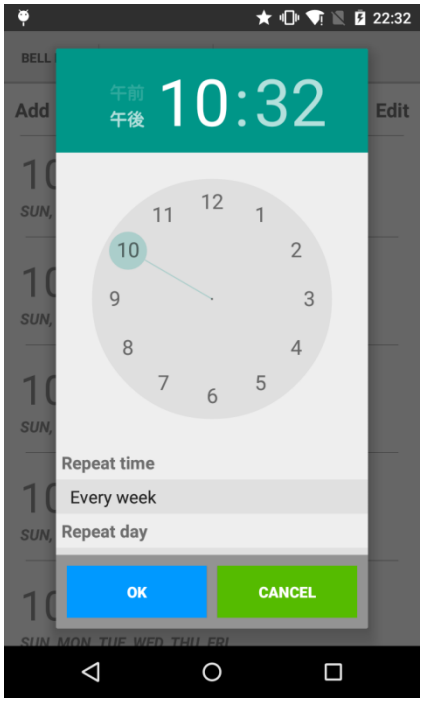


Figure 11:Edit scheduling item.

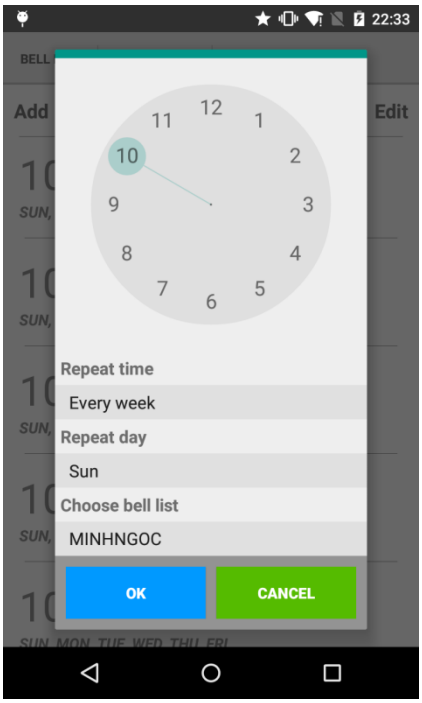


Figure 11:Edit scheduling item.

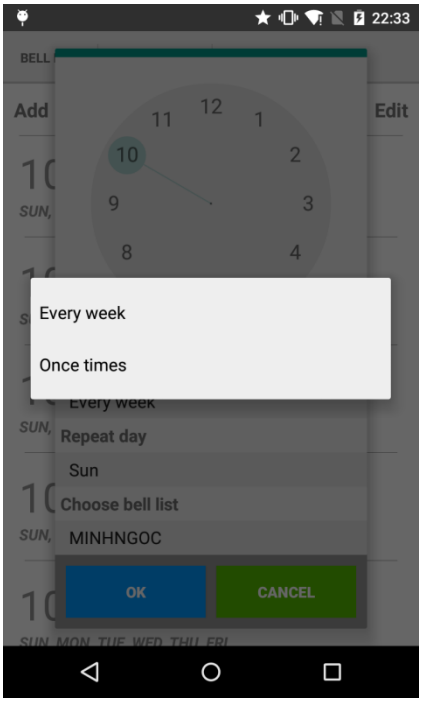


Figure 11:Edit scheduling item.

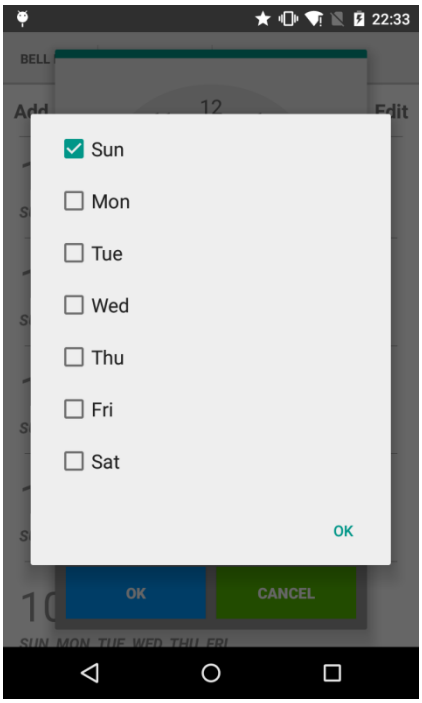


Figure 11:Edit scheduling item.

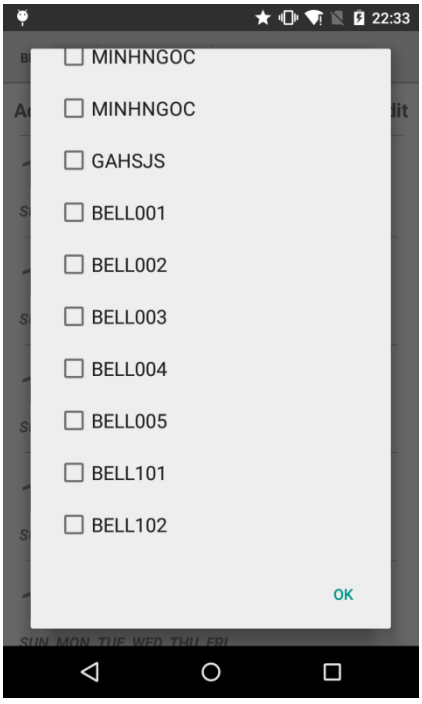


Figure 11:Edit scheduling item.

2.4.11. Remove item

When user clicked on switcher button, the item will be disappear and a remove message will send to belling system.

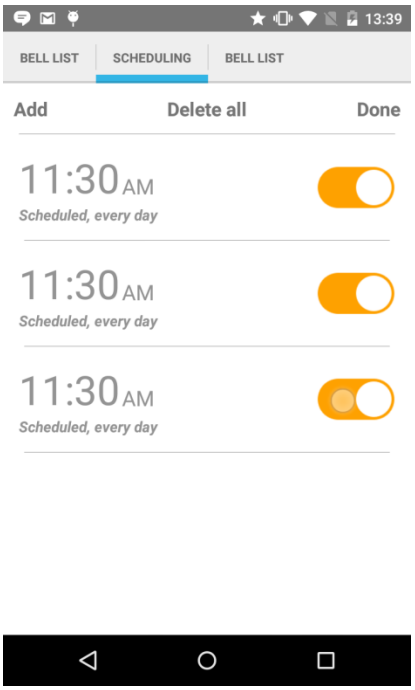


Figure 12:Remove item.

2.4.12. Delete all list schedule

When user delete all schedule, a delete all message will send to belling system

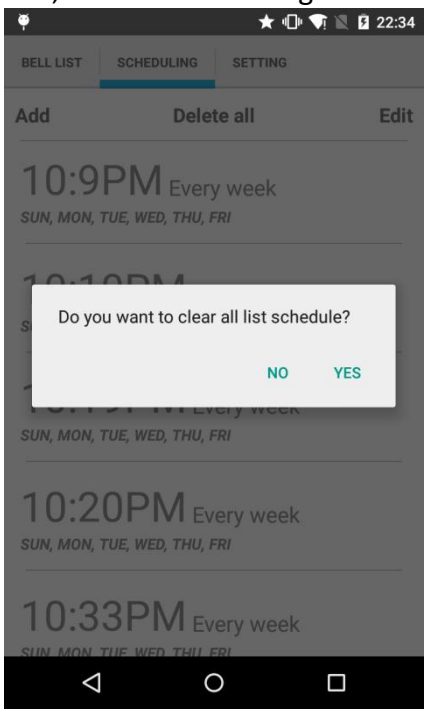


Figure 13:Delete all list schedule.

2.4.13. Setting screen

User sliding to setting screen to authentication with belling system.

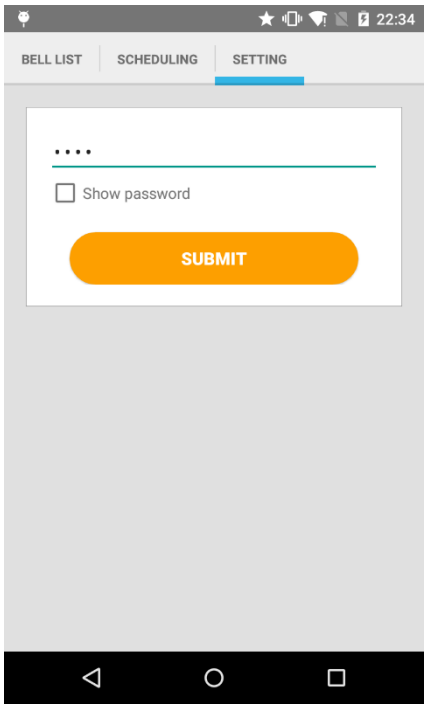


Figure 14:Setting screen.

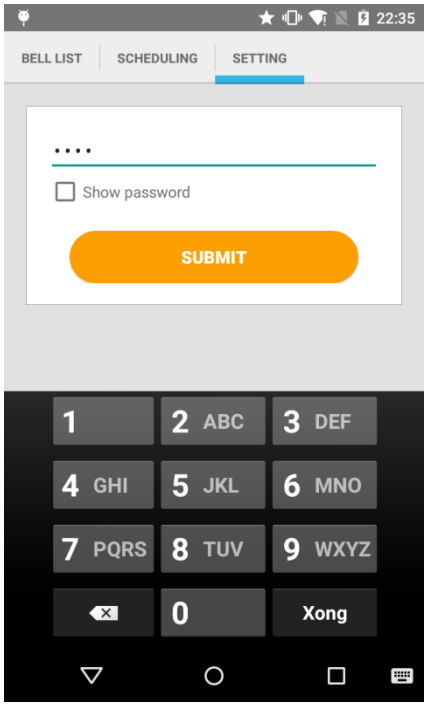


Figure 14:Setting screen.

2.4.14. Show passwrod when input

If not sure about input thing, user could be check on “Show password” check box to show password.

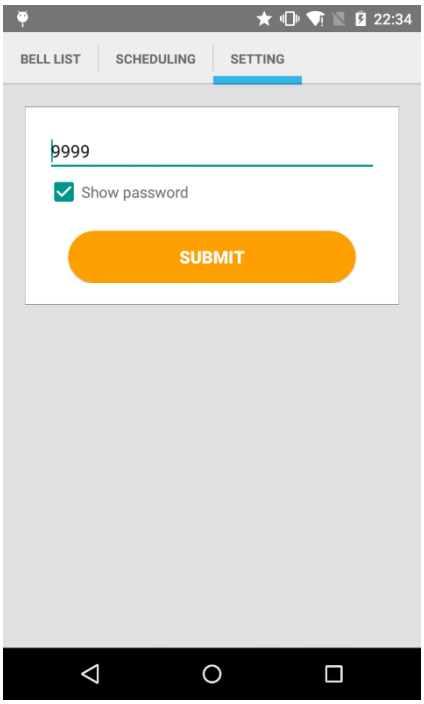


Figure 15:Show password when input.

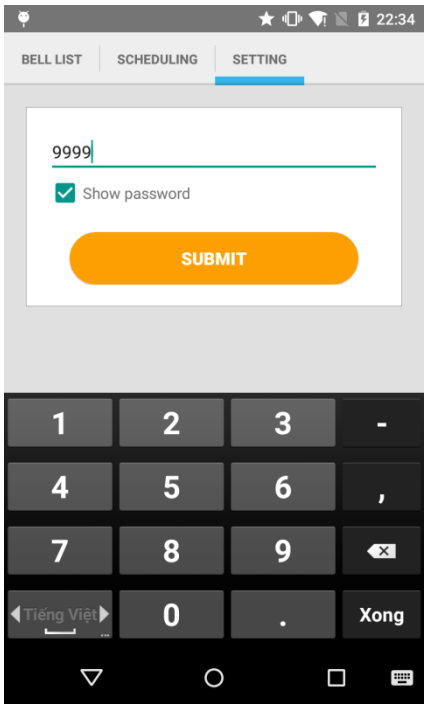


Figure 15:Setting screen.

IV. System Design Document (SDD)

In this section:

- Giving a general description of the functionality, context and design of ASBS (In section 1.)
- Indentifying subsystems, components and the role or responsibility assigned to them. Describing the overall architecture of the system, the architecture of each subsystem and component (in section 2.).
- Describing detailed structure for each component and function followed to main functions. It includes class diagrams, class explanations, and sequence diagrams of the main use cases (in section 3.).
- Describing the functionality of the system from user's perspective. Explaining how the user will be able to use ASBS to complete all the expected features and the feedback information that will be displayed for the user. Giving screenshots which show the interface from the user's perspective (in section 4.)
- Describing the hardware interface of the belling system (in section 5.).
- Describing the structure of data (in section 6.)

1. System Overview

(*Refer to Part I. Introduction - Section 4. System Overview Description*)

2. System Architecture

2.1. Overall System Architecture

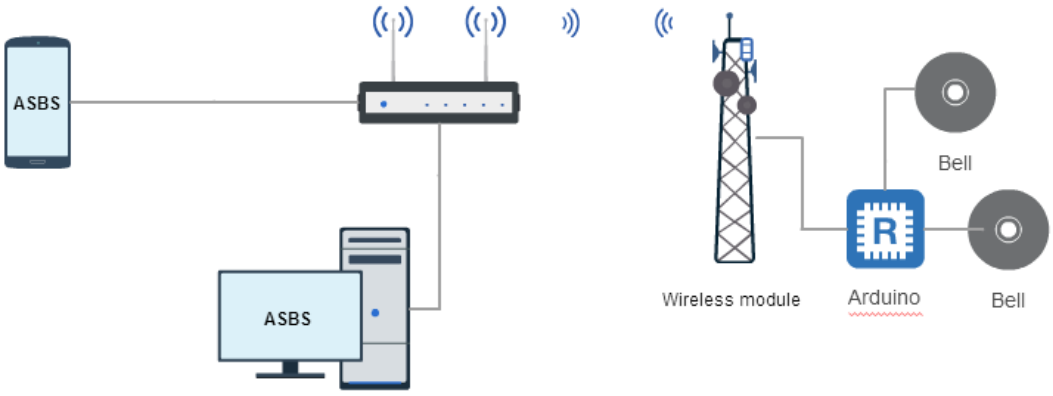


Figure 1: Overall system architecture

No	Entity	Description
1	Smartphone	The user's Smartphone connects to the Belling system via Wi-Fi. To interact with the belling system, the user's device connects to each bulb to provide name (Wiki, Service set 802.11 network) and password of Wi-Fi connection which user's Wi-Fi router provides. The user's Smartphone need installed ASBS Application to control the belling system.
2	Desktop	The user's Desktop connects to the Belling system via Wi-Fi. To interact with the belling system, the user's device connects to each bulb to provide name (Wiki, Service set 802.11 network) and password of Wi-Fi connection which user's Wi-Fi router provides. The user's Desktop need connect to system web to control the belling system.
3	Belling system	The belling system consists of one or many bell group. Each bell group contains 1 bell, 1 Adruino and 1 Wireless module. The Wireless module plays a role as TCP server to receive TCP connection from the user's Smartphone and

		respond. The Wireless module can handle >20 TCP connection. (Hi-Link co. ltd, HK-RM04 User Manual) Adruino module will take the request, process and control bell. Each bell group needs connecting to the user's Wi-Fi router via Wi-Fi before being controlled by The mobile application.
4	Wi-Fi router	The Wi-Fi router plays an important role in the system. The whole system is already in use when the user's Smartphone connect to the Wi-Fi router successfully; the belling system connects to the Wi-Fi router successfully. Thus, all bell group in the belling system is in the same WLAN with the user's Smartphone.

2.2. Mobile Application Architecture

The following diagram shows the architecture of the mobile application:

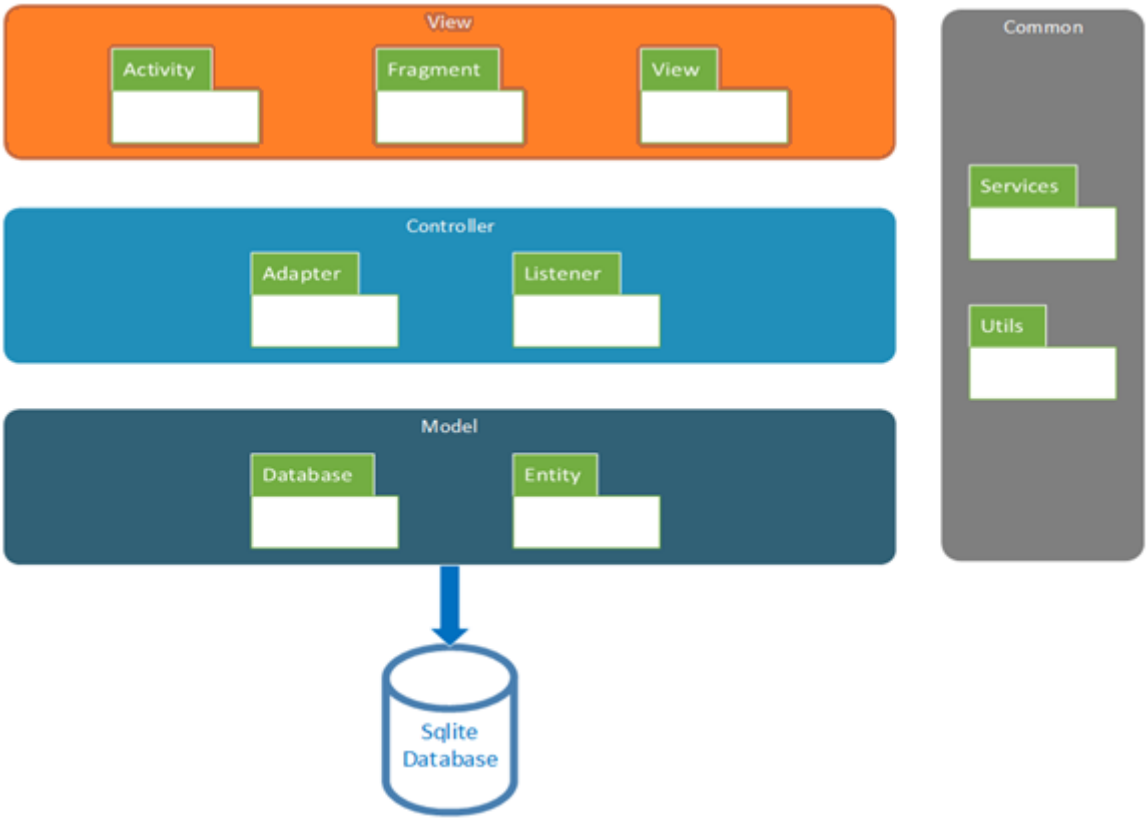


Figure 2:Mobile Application Architecture

- **View**
  - Contains all class activity, fragments and custom widgets are responsible for define the structure, layout and appearance of what display on screen and user can interact with them.

  - The structure of view is define on XML file called is layout file and store in /res folder.
  - Each activity is a screen and it can hold some of fragments which is separation of screen. It helps make a dynamic structure of screen and awesome effect when changing screen.
  - Project using some of custom widgets help application have beautiful design.

- **Controller**
  - The controller can understand as a bridge between data and view on screen. It helps transfer information of model to view display on screen.
  - It helps to retrieves data from model and makes it visible on screen in custom structure of layout like all class Adapter interacts with ListView. The Adapter class help list of item on screen not only have a text but also rich of format with image, change color of text and more…
  - The controller also implements all listener of user's command to class. Android is event-driven programming so controller have a special role and very important.

- **Model**
  - Android is not only event-driven programming but also object-oriented programming. So it use model as class hold all information of separate entity in system.
  - The model also contain class help application interact with SQLite database includes create, insert, edit and get data from database of application. It help application can store static which use in any time unless user uninstall application.

- **Common**
  - Contains all utilities class and services help application send data to bulbs through TCP/IP protocol.

For details, the following diagram shows structure of the mobile application:

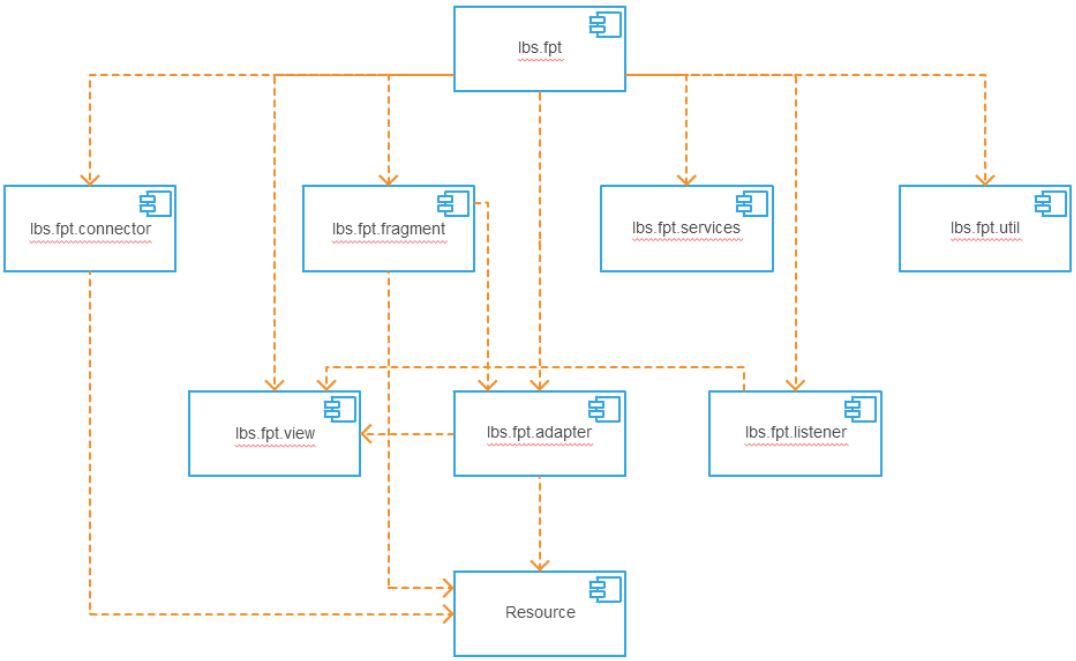


Figure 3: Structure diagram

- Package explanation:

	Package	Description
1	Lbs.fpt	Contains all class activity and main application class.
2	lbs.fpt.fragment	Contains all fragment class using in application.
3	lbs.fpt.services	Contains service class which interact with network and send/receive data from group bell.
4	lbs.fpt.utils	Contains utilities class of application
5	lbs.fpt.view	Contains all class are android widget customized.
6	lbs.fpt.adapter	Contains all adapter class are data binder of all ListView.
7	lbs.fpt.listener	Contains all interface using in application.
8	Resouces	Contains all xml file of project such as layout file of activity, fragment and string.xml define all message using in application.

3. Detail Design

3.1. Control bell system

This part is the detailed design for the following use cases:

- Turn on/off bell remote using mobile application.
- Connect smart phone to Arduino.

- Disconnect smart phone from Arduino.
- View all group bell connect with connected Arduino.
- Authentication with bell system
- Create new schedule
- Edit schedule
- Delete schedule
- Change name of bell (on application).

3.1.1 Class Diagram

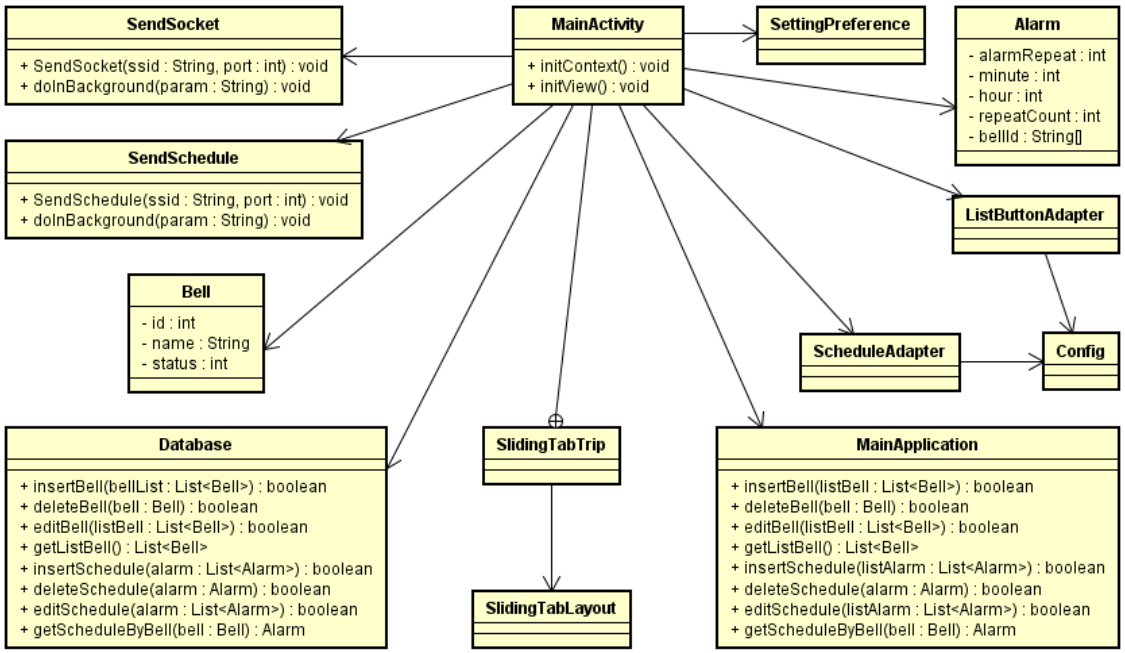

SendSocket
+ SendSocket(ssid : String, port : int) : void
+ doInBackground(param : String) : void
MainActivity
+ initContext() : void
+ initView() : void
SettingPreference
Alarm
- alarmRepeat : int
- minute : int
- hour : int
- repeatCount : int
- bellId : String[]
SendSchedule
+ SendSchedule(ssid : String, port : int) : void
+ doInBackground(param : String) : void
ListButtonAdapter
Bell
- id : int
- name : String
- status : int
ScheduleAdapter
Config
Database
+ insertBell(bellList : List<Bell>) : boolean
+ deleteBell(bell : Bell) : boolean
+ editBell(listBell : List<Bell>) : boolean
+ getListBell() : List<Bell>
+ insertSchedule(alarm : List<Alarm>) : boolean
+ deleteSchedule(alarm : Alarm) : boolean
+ editSchedule(alarm : List<Alarm>) : boolean
+ getScheduleByBell(bell : Bell) : Alarm
SlidingTabTrip
SlidingTabLayout
MainApplication
+ insertBell(listBell : List<Bell>) : boolean
+ deleteBell(bell : Bell) : boolean
+ editBell(listBell : List<Bell>) : boolean
+ getListBell() : List<Bell>
+ insertSchedule(listAlarm : List<Alarm>) : boolean
+ deleteSchedule(alarm : Alarm) : boolean
+ editSchedule(listAlarm : List<Alarm>) : boolean
+ getScheduleByBell(bell : Bell) : Alarm

Figure 1:Class diagram

3.1.2 Class Diagram Explanation

3.1.2.1 Class Database

This class is help mobile interact with SQLite database of application. It store information of connected bulbs.

- *Attributes:*
  N/A
- *Operations:*
  boolean insertBell(List<Bell> bell):
  Insert list new bell to list of connected group bell.

  Parameters and return

	Name	***Type***	***Description***
Parameter	bell	List<Bell>	List of bell which user want to add to list of connected group bell.

Return		boolean	Indicate which insert a new bell success or not. Return TRUE if insert a new bell success and FALSE in otherwise.

boolean deleteBell(Bell bell):
Delete a bell from list of connection group bell.

Parameters and return

	Name	***Type***	***Description***
Parameter	bell	Bell	Bell which user wants to delete from list of connected bell.
Return		boolean	Indicate which delete bell success or not. Return TRUE if deleted success and FALSE in otherwise.

boolean editBell(List<Bell> bell):
Modify information of a bell.

Parameters and return

	Name	***Type***	***Description***
Parameter	bell	**List<Bell>**	List bell which holds modified information. Bell will be modify by it's bell id.
Return		boolean	Indicate which edit bell success or not. Return TRUE if deleted success and FALSE in otherwise.

List<Bell> getListBell():
Get list of connected bell.

Parameters and return

	Name	***Type***	***Description***
Parameter	N/A	N/A	N/A
Return		List<Bell>	Return list of connected bell from database.

boolean insertSchedule(List<Alarm> alarm):
Insert list new alarm to list of connected alarms.

Parameters and return

	Name	***Type***	***Description***
Parameter	alarm	List<Alarm>	List of alarm which user want to add to list of connected alarms.
Return		boolean	Indicate which insert a new alarm success or not. Return TRUE if insert a new bell success and FALSE in otherwise.

boolean deleteSchedule(Alarm alarm):
Delete a alarm from list of connection alarms.

Parameters and return

	Name	***Type***	***Description***
Parameter	alarm	Alarm	Alarm which user wants to delete from list of connected alarms.
Return		boolean	Indicate which delete alarm success or not. Return TRUE if deleted success and FALSE in otherwise.

boolean editSchedule(List<Alarm> alarm):
Modify information of a alarm.

Parameters and return

	Name	***Type***	***Description***
Parameter	alarm	List<Alarm>	List alarm which holds modified information. Alarm will be modify by it's bell id.
Return		boolean	Indicate which edit alarm success or not. Return TRUE if deleted success and FALSE in otherwise.

Alarm getScheduleByBell(Bell bell):
Get list of connected alarm.

Parameters and return

	Name	***Type***	***Description***
Parameter	bell	Bell	Bell which want to get schedule information of it.
Return		Alarm	Return the information alarm of bell.

3.1.2.2 MainApplication

This class holds static data which use for whole application. That data can be accessed in any Activity, Service

- *Attributes:*
  N/A
- *Operations:*
  boolean insertBell(List<Bell> bell):
  Insert list new bell to list of connected bell. List of connected bell will be updated in database.

Parameters and return

	Name	***Type***	***Description***
Parameter	bell	List<Bell>	List of bell which user wants to add to list of connected bell.
Return		boolean	Indicate which insert a new bell success or not. Return TRUE if insert a new bulb success and FALSE in otherwise.

boolean deleteBell(Bell bell):
Delete a bell from list of connected bell. List of connected bell will be updated into database also.

Parameters and return

	Name	***Type***	***Description***
Parameter	bell	Bell	Bell which user wants to delete from

			list of connected bell.
Return		boolean	Indicate which delete bell success or not. Return TRUE if deleted success and FALSE in otherwise.

boolean editBell(List<Bell> bell):
Modify information of a bell. List of connected bell will be updated in database.

Parameters and return

	Name	***Type***	***Description***
Parameter	bell	**List<Bell>**	List bell which holds modified information. Bell will be modify by it's bell id.
Return		boolean	Indicate which edit bell success or not. Return TRUE if deleted success and FALSE in otherwise.

List<Bell> getListBell():
Get list of connected bell. List of connected bell will be updated in database.

Parameters and return

	Name	***Type***	***Description***
Parameter	N/A	N/A	N/A
Return		List<Bell>	Return list of connected bell from database.

boolean insertSchedule(List<Alarm> alarm):
Insert list new alarm to list of connected alarms. List of alarm will be updated in database.

Parameters and return

	Name	***Type***	***Description***
Parameter	alarm	List<Alarm>	List of alarm which user want to add to list of connected alarms.
Return		boolean	Indicate which insert a new alarm

			success or not. Return TRUE if insert a new bell success and FALSE in otherwise.

boolean deleteSchedule(Alarm alarm):
Delete a alarm from list of connection alarms. List of alarm will be updated in database.

Parameters and return

	Name	***Type***	***Description***
Parameter	alarm	Alarm	Alarm which user wants to delete from list of connected alarms.
Return		boolean	Indicate which delete alarm success or not. Return TRUE if deleted success and FALSE in otherwise.

boolean editSchedule(List<Alarm> alarm):
Modify information of a alarm. List of alarm will be updated in database.

Parameters and return

	Name	***Type***	***Description***
Parameter	alarm	List<Alarm>	List alarm which holds modified information. Alarm will be modify by it's bell id.
Return		boolean	Indicate which edit alarm success or not. Return TRUE if deleted success and FALSE in otherwise.

Alarm getScheduleByBell(Bell bell):
Get list of connected alarm. List of alarm will be updated in database.

Parameters and return

	Name	***Type***	***Description***
Parameter	bell	Bell	Bell which want to get schedule information of it.

Return		Alarm	Return the information alarm of bell.

3.1.2.3 MainActivity

This class is Activity for bell controller.

- *Attributes:*
  N/A
- *Operations:*

void initContext():
Initiate context variables for activity such as Context, Activity, list of connected bell from database.

Parameters and return

	Name	***Type***	***Description***
Parameter	N/A	N/A	N/A
Return		void	

void initView():
Initiate main layout of activity.

Parameters and return

	Name	***Type***	***Description***
Parameter	N/A	N/A	N/A
Return		void	

3.1.2.4 SendSocket

Class extend from AsyncTask using for send command to bell while setup a new bell.

- *Attributes:*
  N/A
- *Operations:*

void SendSocket(String ssid, int port):
Constructor of class.

Parameters and return

	Name	***Type***	***Description***
Parameter	ssid	String	IP address of bell which want to setup.
	port	int	Port of connection (9119).

Return		void	

boolean doInBackground(String param):
Method run while send command to bell.

Parameters and return

	Name	***Type***	***Description***
Parameter	param	String	Command which send to bell.
Return		boolean	TRUE if send command success and FALSE in otherwise.

3.1.2.5 SendSchedule

Class extend from AsyncTask using for send command to bell system while setup a new schedule.

- *Attributes:*
  N/A
- *Operations:*

void SendSchedule(String ssid, int port):
Constructor of class.

Parameters and return

	Name	***Type***	***Description***
Parameter	ssid	String	IP address of bell which want to setup.
	port	int	Port of connection (9119).
Return		void	

boolean doInBackground(String param):
Method run while send command to bell.

Parameters and return

	Name	***Type***	***Description***
Parameter	param	String	Command which send to bell.
Return		boolean	TRUE if send command success and FALSE in otherwise.

3.1.2.6 Bell

Bell class using to hold bell information.

- *Attributes:*
  int id: Identify of the bell.

  String name: Name of the bell

  String status: Status of the bell

- *Operations:*
  String getId():

  Parameters and return

	Name	***Type***	***Description***
Parameter	N/A	N/A	N/A
Return		String	Id of the bell

void setID(String id):

Parameters and return

	Name	***Type***	***Description***
Parameter	Id	String	Id of the bell
Return	N/A	N/A	N/A

String getName():

Parameters and return

	Name	***Type***	***Description***
Parameter	N/A	N/A	N/A
Return		String	Name of the bell

void setName(String name):

Parameters and return

	Name	***Type***	***Description***
Parameter	name	String	Name of the bell
Return	N/A	N/A	N/A

STATUS getStatus():

Parameters and return

	Name	*Type*	*Description*
Parameter	N/A	N/A	N/A
Return		STATUS	Status of the bell

void setStatus(STATUS status):

Parameters and return

	Name	*Type*	*Description*
Parameter	status	STATUS	Status of the bell
Return	N/A	N/A	N/A

3.1.2.7 STATUS

- *Attributes:*
  RINGING: the ringing status of bell

  PREPARING: the preparing status of bell

  STOPING: the stop status of bell

  DISABLE: the disable status of bell

- *Operations:*
  N/A

3.1.2.8 Alarm

The class constain the information of alarm.

- *Attributes:*
  int alarmRepeat: Repeat times of the alarm.

  int minute: Start ringing time

  int hour: Start ringing time

  int repeatCount: Times of repeat

  String[] bellId: List bell id in this alarm schedule

- *Operations:*
  void setAlarmRepeat(int alarmRepeat):

Parameters and return

	Name	***Type***	***Description***
Parameter	alarmRepeat	int	Set the repeat times of the alarm
Return	N/A	N/A	N/A

int getAlarmRepeat ():

Parameters and return

	Name	***Type***	***Description***
Parameter	N/A	N/A	N/A
Return		int	Get the repeat times of the alarm

void setMinute(int minute):

Parameters and return

	Name	***Type***	***Description***
Parameter		int	Set the start ringing time
Return	N/A	N/A	N/A

int getMinute():

Parameters and return

	Name	***Type***	***Description***
Parameter	N/A	N/A	N/A
Return		int	Get the start ringing time

void setHour(int hour):

Parameters and return

	Name	***Type***	***Description***
Parameter	hour	int	Set the start ringing time
Return	N/A	N/A	N/A

int getHour ():

Parameters and return

	Name	*Type*	*Description*
Parameter	N/A	N/A	N/A
Return		int	Get the start ringing time

void setRepeatCount(int repeatCount):

Parameters and return

	Name	*Type*	*Description*
Parameter	repeatCount	int	Set the times of repeat
Return	N/A	N/A	N/A

int getRepeatCount():

Parameters and return

	Name	*Type*	*Description*
Parameter	N/A	N/A	N/A
Return		int	Get the times of repeat

void setListBellId(String[] bellIdList**)**:

Parameters and return

	Name	*Type*	*Description*
Parameter	listBellId	String[]	Set list bell id in this alarm schedule
Return	N/A	N/A	N/A

String[] getAlarmRepeat ():

Parameters and return

	Name	*Type*	*Description*
Parameter	N/A	N/A	N/A
Return		String[]	Get list bell id in this alarm schedule

3.1.2.9 ListButtonAdapter

Adapter for list button list view. This adapter store data and render view for list button view.

- *Attributes:*

Context context: The context to inflate view.

List<Button> listButton: List button data

IOnButtonClickedListener onButtonClickedListener: Listener for button clicked.

- *Operations:*
  N/A

3.1.2.10 Interface IOnButtonClickedListener

This interface implement when user taps into button bell.

- *Attributes:*
  N/A
- *Operations:*

void onButtonClicked(): On the button clicked. This method is called when user tap to button bell.

Parameters and return

	Name	***Type***	***Description***
Parameter	N/A	N/A	N/A
Return	N/A	N/A	N/A

3.1.2.11 ScheduleAdapter

This class is binder to bind list of ListView display list of list Bell button on screen with custom style.

- *Attributes:*
  N/A
- *Operations:*
  N/A

3.1.2.12 SlidingTabLayout

This class initial the layout of the tab screen.

- *Attributes:*
  N/A
- *Operations:*
  N/A

3.1.2.13 SlidingTabTrip

This class initial the layout of the tab title and control the animation of tab changed.

- *Attributes:*
  N/A
- *Operations:*
  N/A

3.1.2.14 Config

This class contain the configuration of this application.

- *Attributes:*
  static final boolean IS_DEBUG: The status of the application is debuging or not.

- *Operations:*
  N/A

3.1.2.15 SettingPreference

This class contain the setting of this application. That is an local database store in xml file.

- *Attributes:*
  final String KEY_IS_GET_NOTIFICATION: XML key for "is get notification or not" data.

  final String KEY_IS_NOTIFICATION_NEXT_RINGING: XML key for "is get notification next ringing" data.

  final String KEY_IS_GET_NOTIFICATION_FROM_OTHER: XML key for "is get notification from another device or not" data.

- *Operations:*

SettingPreference getInstance(): Singleton pattern for getting up the preference class.

Parameters and return

	Name	***Type***	***Description***
Parameter	N/A	N/A	N/A
Return		SettingPreference	Get the instance of this XML class

SharedPreference getSharedPreference(): Get the shared preference from android.

Parameters and return

	Name	***Type***	***Description***
Parameter	N/A	N/A	N/A

Return		SharedPreference	Get the shared preference from android

Editor getEditor(): Get editor class from shared peference for write the new data to the XML file.

Parameters and return

	Name	***Type***	***Description***
Parameter	N/A	N/A	N/A
Return		Editor	Get editor class from shared peference for write the new data to the XML file

void saveGetNotification(boolean isNotification): Save the flag "get notification" to the xml file.

Parameters and return

	Name	***Type***	***Description***
Parameter	isNotification	boolean	The flag "get notification" to save to the xml file.
Return	N/A	N/A	N/A

boolean isGetNotification(): Get the flag "get notification" from the xml file.

Parameters and return

	Name	***Type***	***Description***
Parameter	N/A	N/A	N/A
Return		boolean	Get the flag "get notification" from the xml file.

void saveNotificationNextRinging(boolean isNotification): Save the flag "is get notification next ringing" to the xml file.

Parameters and return

	Name	***Type***	***Description***
Parameter	isNotification	boolean	The flag "is get notification next

			ringing” to save to the xml file.
Return	N/A	N/A	N/A

boolean isGetNotificationNextRinging():Get the flag “is get notification next ringing” from the xml file.

Parameters and return

	Name	***Type***	***Description***
Parameter	N/A	N/A	N/A
Return		boolean	Get the flag “is get notification next ringing” from the xml file.

Return		boolean	Get the flag “get notification” from the xml file.

void saveNotificationFromAnother(boolean isNotification): Save the flag “is get notification from another” to the xml file.

Parameters and return

	Name	***Type***	***Description***
Parameter	isNotification	boolean	The flag “is get notification from another” to save to the xml file.
Return	N/A	N/A	N/A

boolean isGetNotificationFromAnothe (): Get the flag “is get notification next ringing” from the xml file.

Parameters and return

	Name	***Type***	***Description***
Parameter	N/A	N/A	N/A
Return		boolean	Get the flag “is get notification from another” from the xml file.

3.1.2.16 MainActivity

This class is the activity control the bell.

- *Attributes:*

N/A

- *Operations:*

  void intContext():Initiate context variables for activity such as Context, Activity, list of connected bulbs from database.

  Parameters and return

	Name	***Type***	***Description***
Parameter	N/A	N/A	N/A
Return	N/A	N/A	N/A

  void initView(): Initial the activity main view.

  Parameters and return

	Name	***Type***	***Description***
Parameter	N/A	N/A	N/A
Return	N/A	N/A	N/A

3.1.3 Sequence Diagram

3.1.3.1 View all bell connect

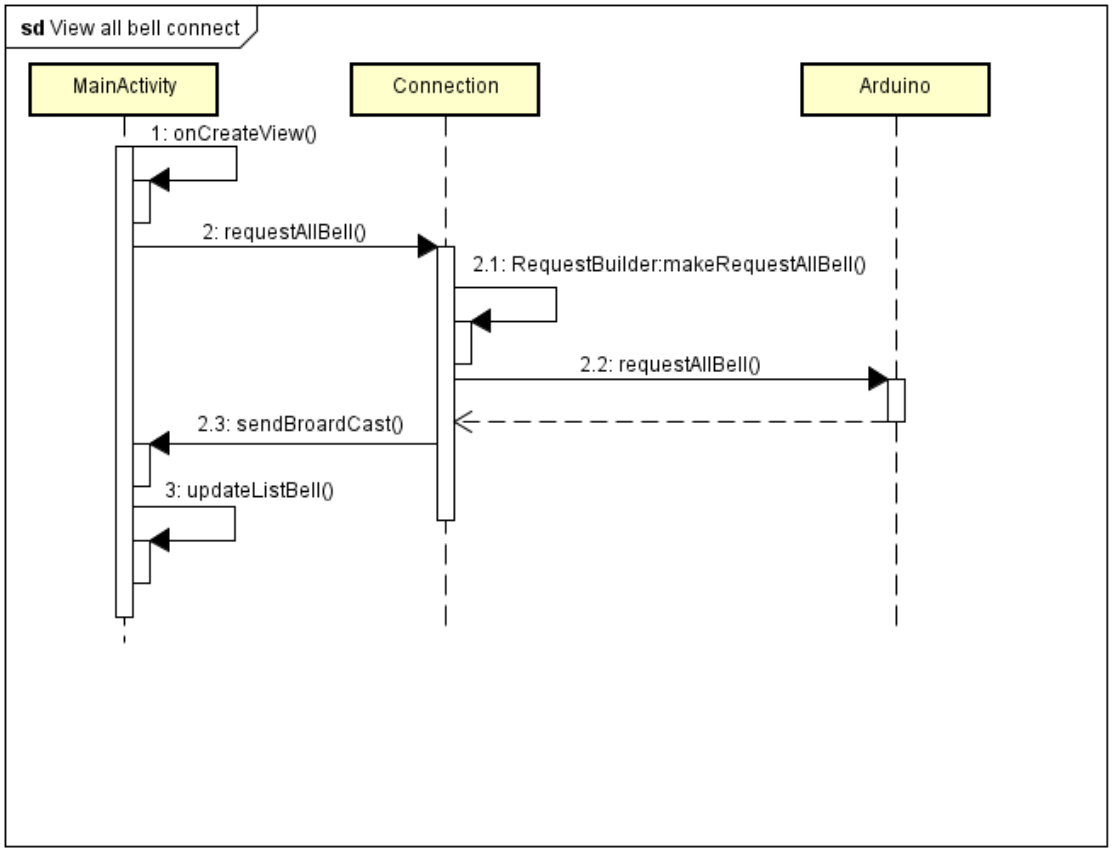


Figure 2: View all bell connect

3.1.3.2 Turn on bell remote using mobile application

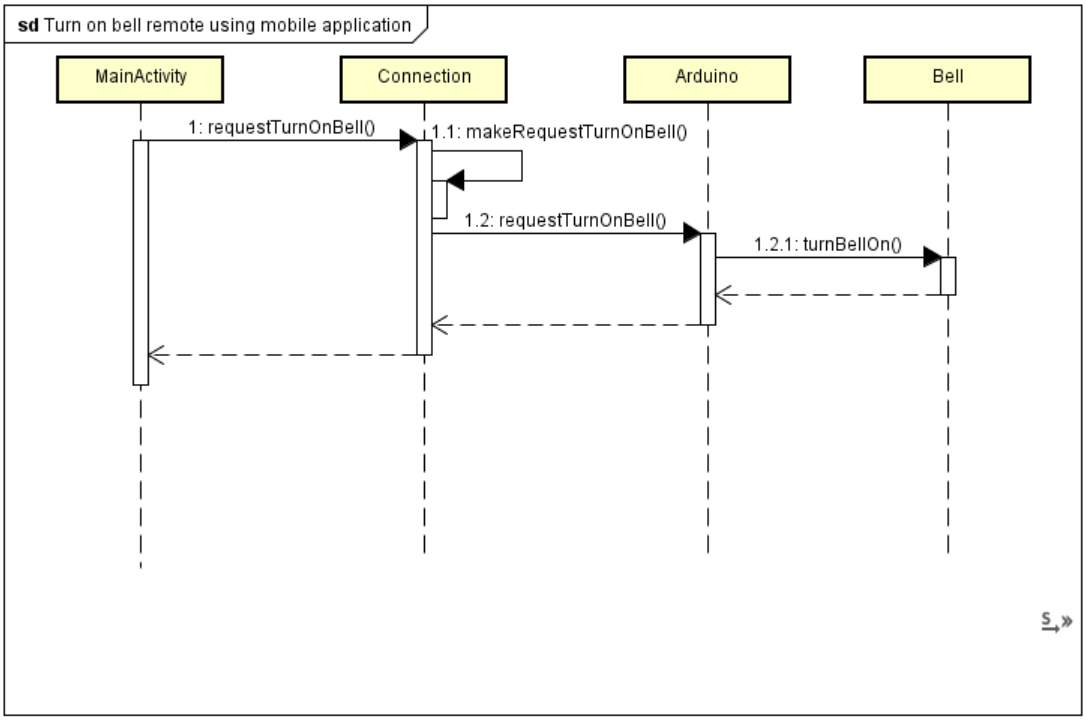


Figure 3: Turn on bell remote using mobile application

3.1.3.3 Turn off bell remote using mobile application

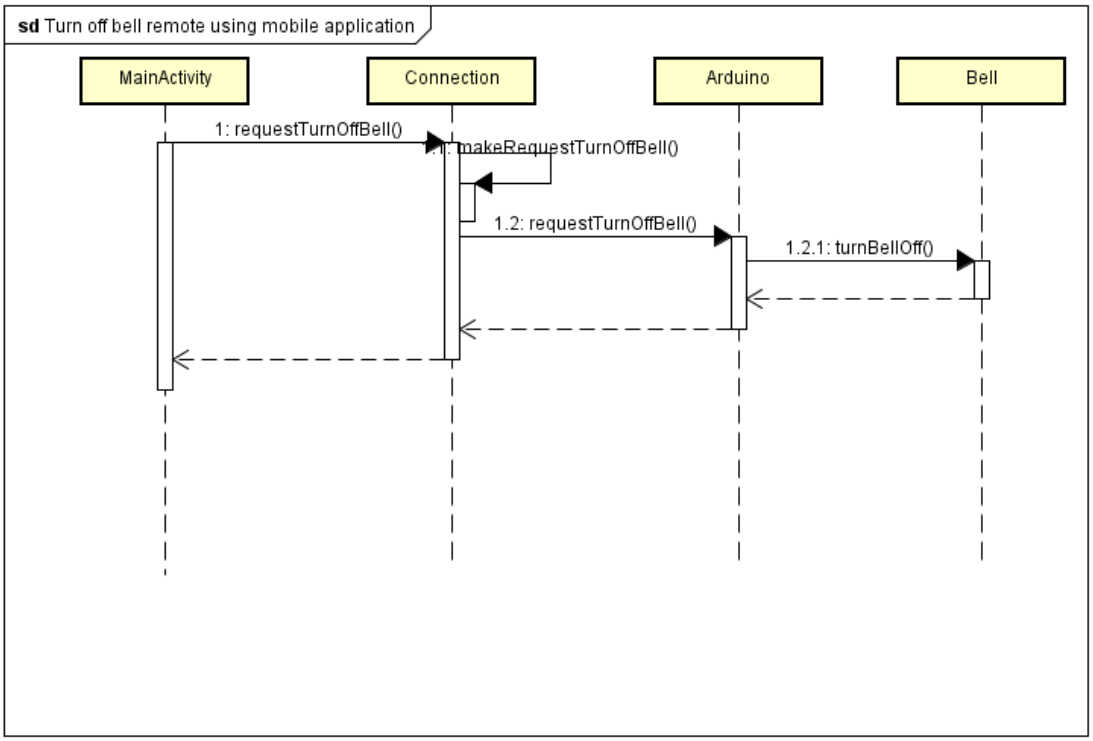


Figure 4:Turn off bell remote using mobile application

3.1.3.4 Change name of bell

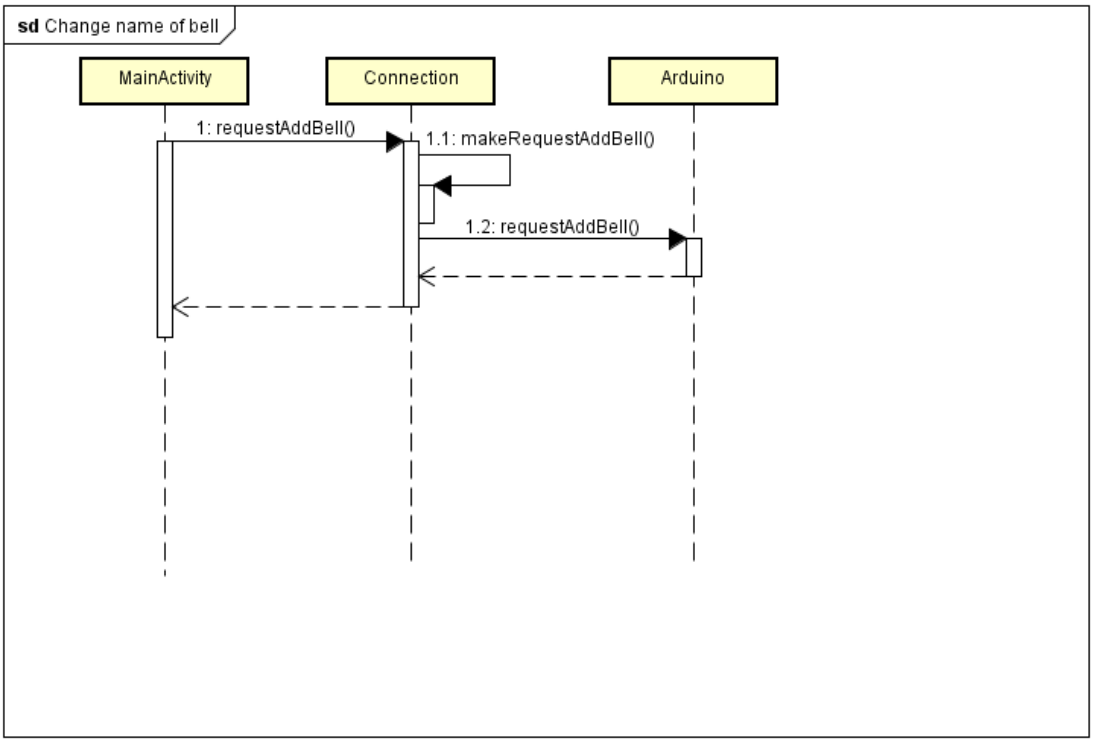


Figure 5: Change name of bell

3.1.3.5 Set alarm for bell

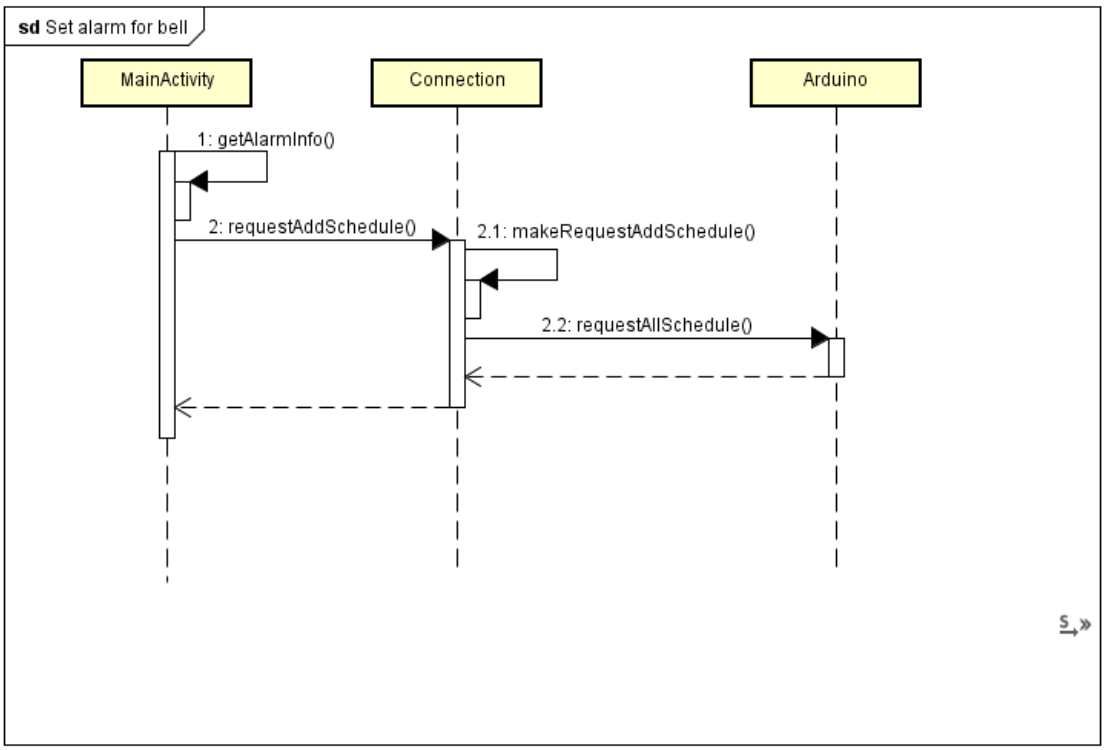


Figure 6: Set alarm for bell

3.1.3.6 Delete all alarm

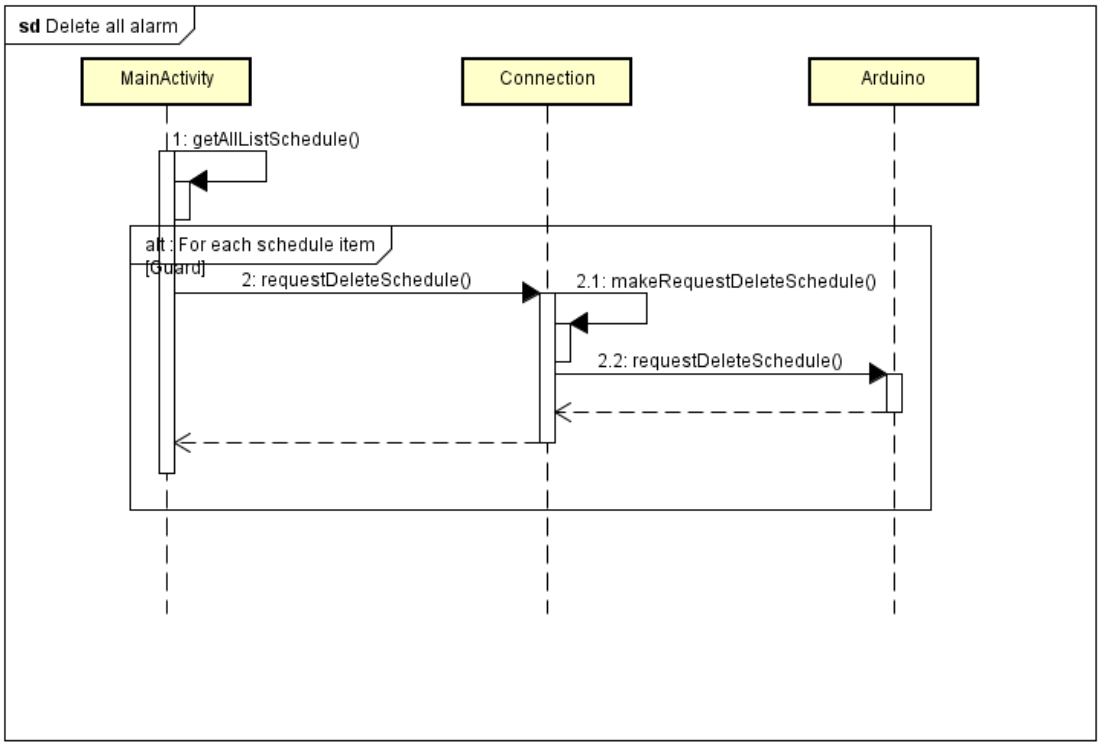


Figure 7: Delete all alarm

3.1.3.7 Delete selected alarm

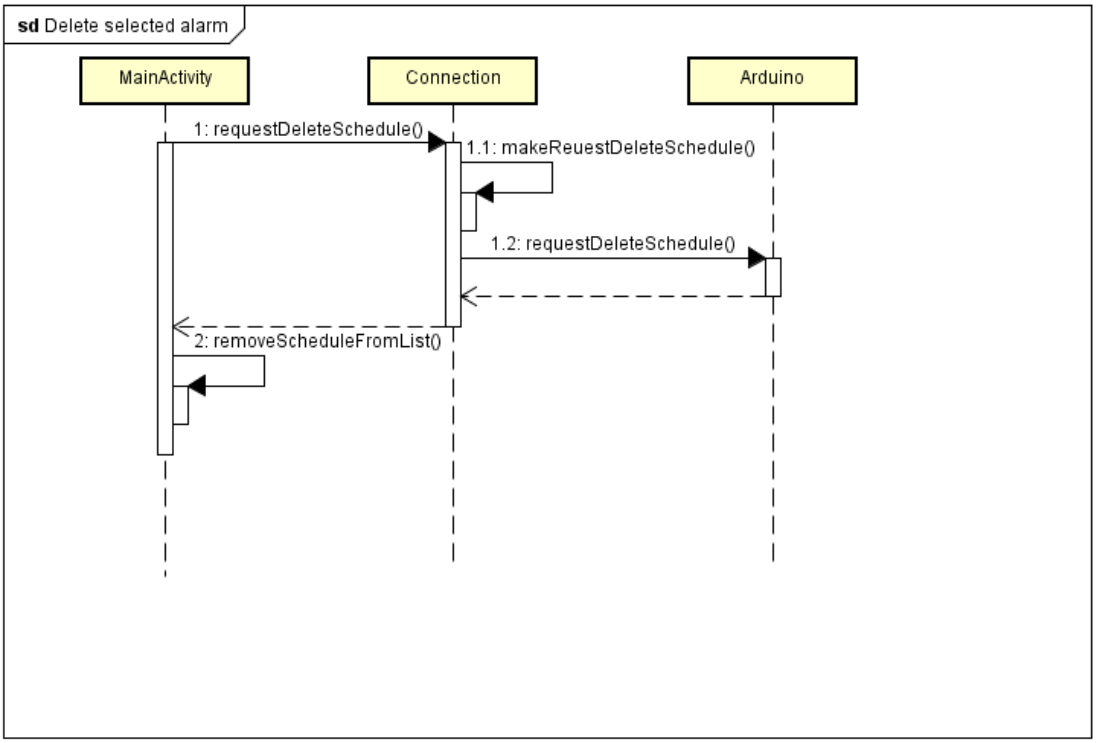


Figure 8: Delete selected alarm

3.1.3.8 Change password

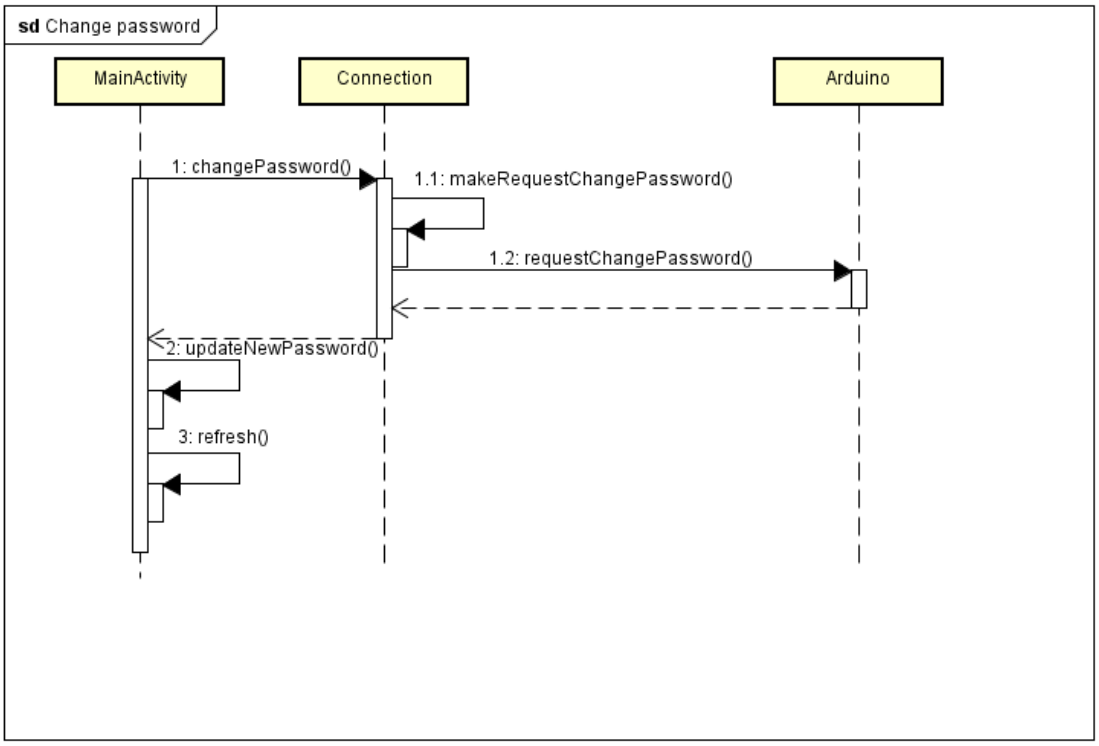


Figure 9: Change password

4. User Interface Design

(Reference to Prototype)

5. Hardware Interface Design

5.1. Hardware description

The Auto Bell is based on the Arduino Pro mini with an integrated Wi-Fi module and 3 super bell.
The following illustration shows the wired circuit diagram and schematic diagram of the Automatic school bell system.

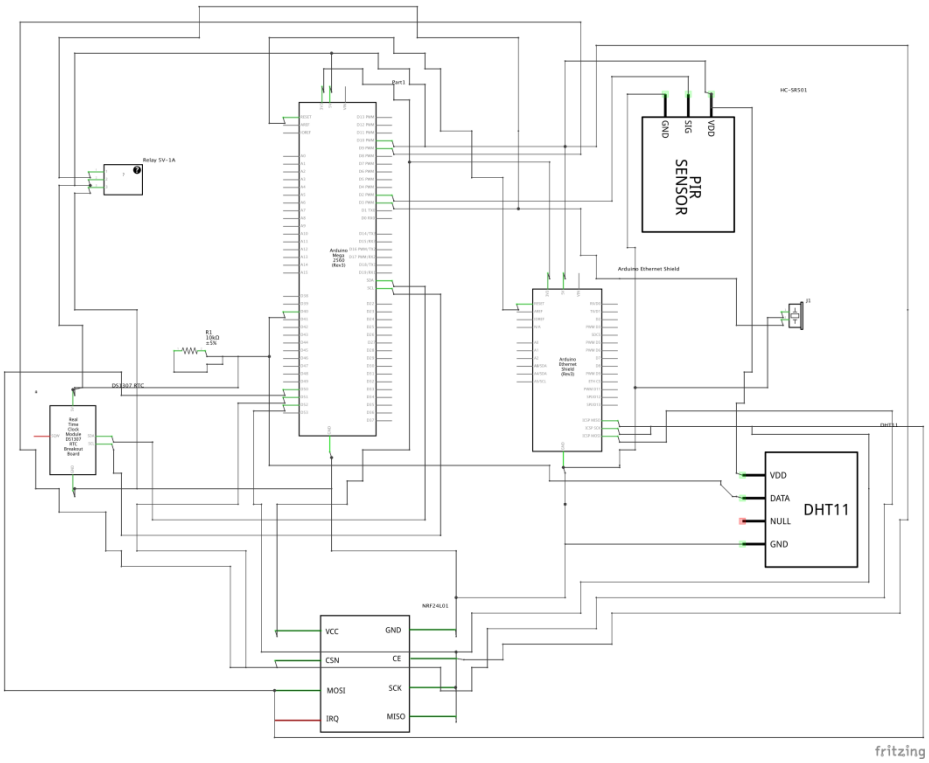


Figure 10: Schematic Diagram- Master

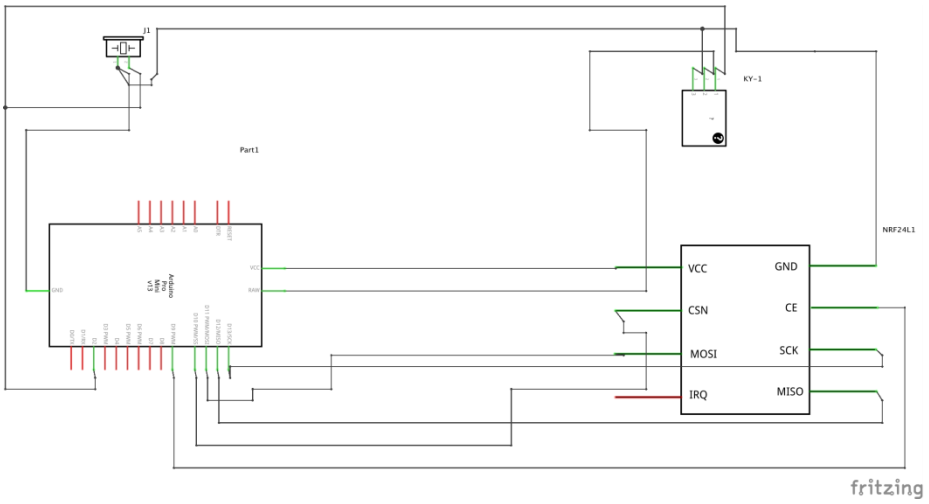


Figure 11: Schematic Diagram - Client

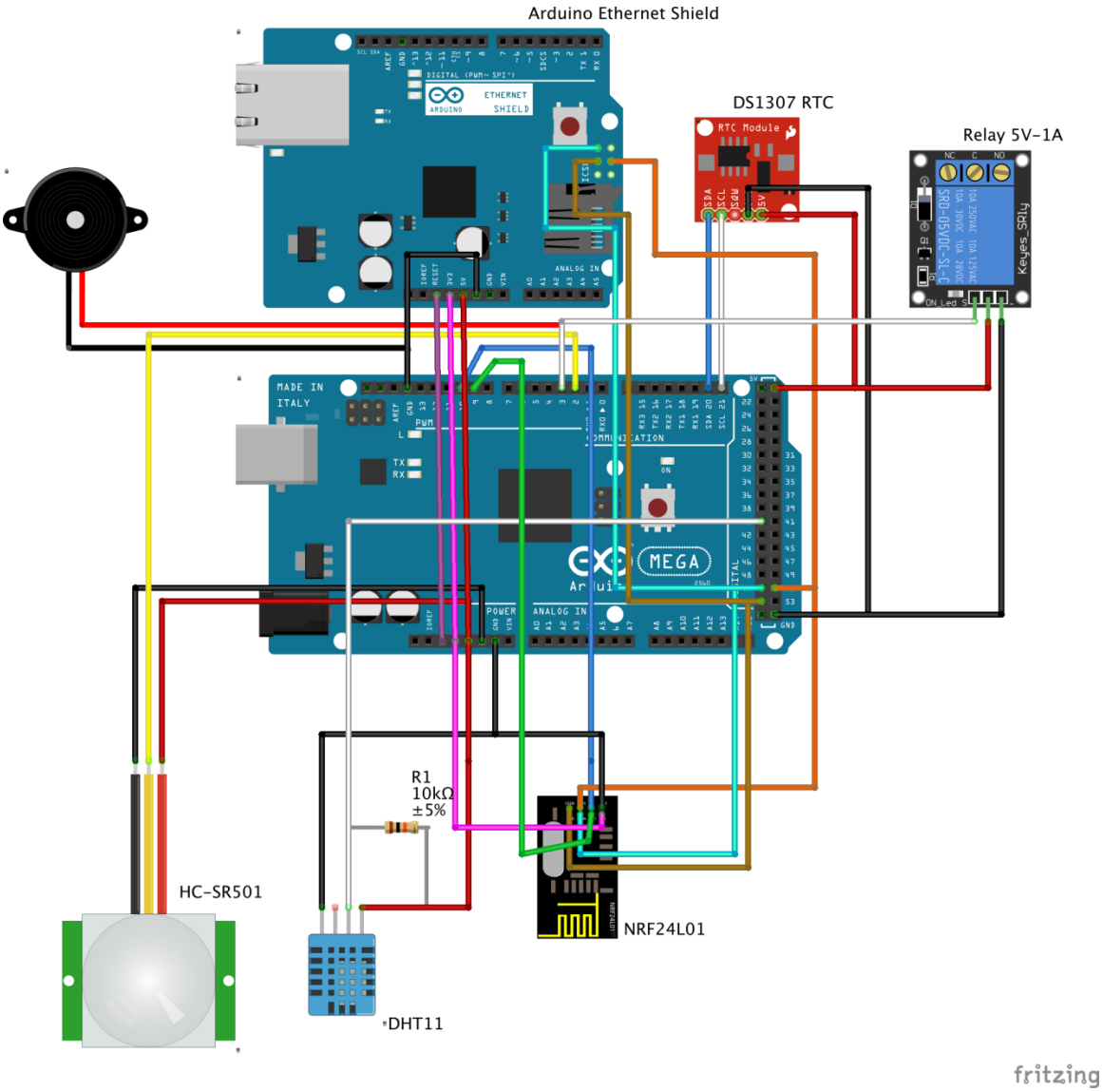


Figure 12: ASBS's Wired circuit diagram - Master

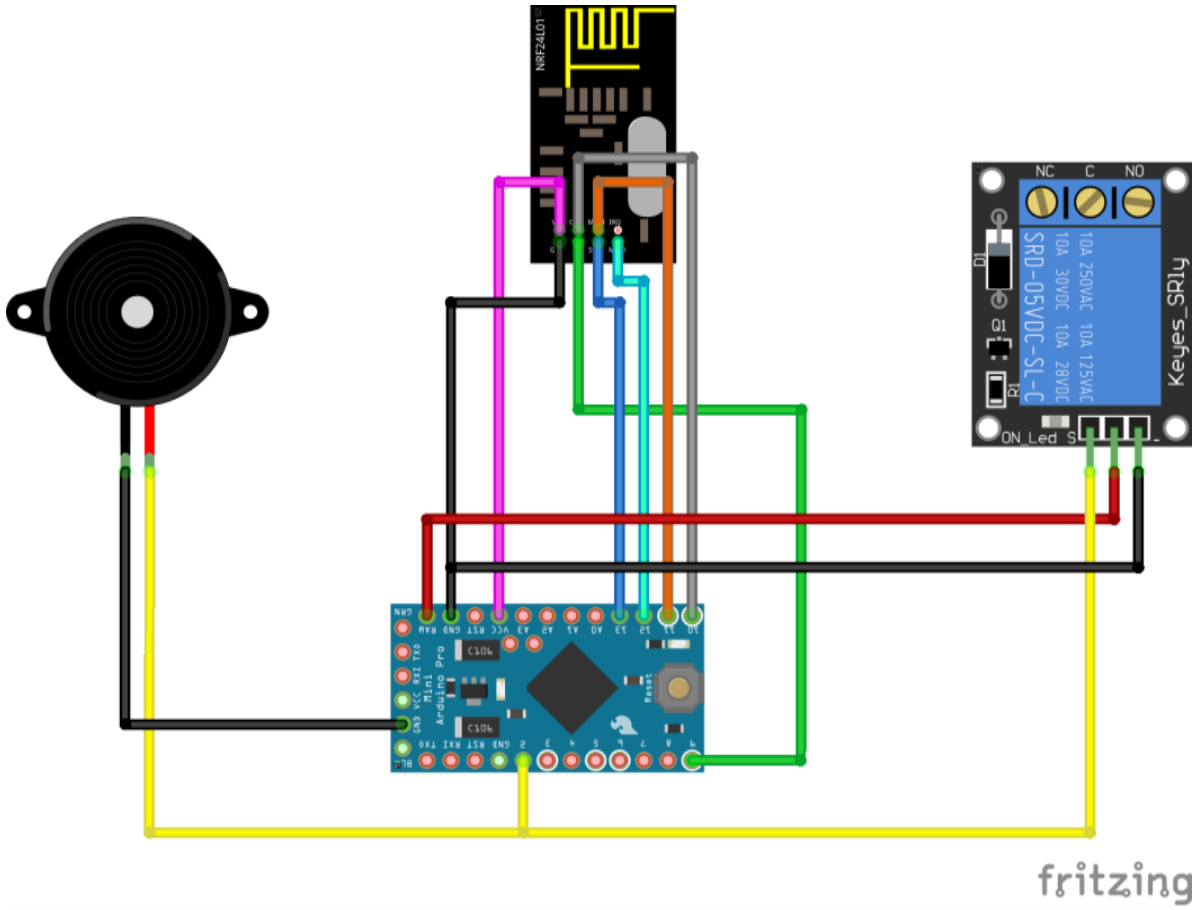


Figure 13: ASBS's Wired circuit diagram - Client

5.2. Hardware component

Arduino Pro mini	
Position	Main center processing unit
Description	Arduino Pro Mini uses Atmega328Pchipwhich is an open-source physical computing platform. Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of bell, motors, and other physical outputs. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing).
Duty	Get data from mobile application and process it depending on what type of data it got and send feedback to mobile.
Specification	• Operating Voltage: 5V • Input Voltage (recommended): 7-12V

	• Input Voltage (limit): 6-20V • Digital I/O Pins: 14 • PWM Digital I/O Pins: 6 • Analog Input Pins: 6 • DC Current per: I/O Pin 40 mA DC Current for: 3.3V Pin 50 mA • Flash Memory: 32 KB • Flash Memory for Bootloader: 0.5 KB • SRAM: 2 KB • EEPROM: 1 KB • Clock Speed: 16 MHz

5.3. Communication protocol – ASBS Command

In this section, communication protocol between smart phone and ASBS via self-defined string called "ASBS command" will be defined in the following table:

ASBS command	
/<whois>	Check is that belling system or not
/<setSchedule>/<hour>/<minute>/<alarmRepeat>/<repeatCount>/<effectedBellCount>/<effectedBells>	Update schedule
/<getSchedules>	Get schedule
/<getTime>	Get current time
/<settime>/<second>/<minute>/<hour>/<date>/<month>/<year>	Set new time
/<getBells>	Get the list bell
/<addBell>/<bellID>/<bellName>	Add a new bell to system
/<removeBell>/<bellID>	Remove a bell from system

More detail
Check is that belling system or not
Function: Check is that belling system or not **Format:** /<whois> **Parameters:** **whois**: Api. **Return:** automatic school bell system
Update schedule
Function: Update schedule

Format:
/<setSchedule>/<hour>/<minute>/<alarmRepeat>/<repeatCount>/<effectedBellCount>/<effectedBells>
Parameters:
setSchedule: Api.
alarmRepeat:

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
sat	fri	thu	wed	tue	mon	sun	ignore this bit

minute: minute
hour: hour
repeatCount:
+ 0: if this schedule is weekly repeat
+ 1: if this schedule is triggered only one time
effectedBellCount: month
the number of bell which this schedule effected
effectedBells:
string of bell ids separated by / which is effected by this schedule
Return:
ok if success, **ERROR** if failure.

Get schedule

Function:
Get schedule
Format:
/<getSchedules>
Parameters:
getSchedules: Api.
Return:
All the current schedules line by line
each line is a schedule with format: hour|minute|alarmRepeat|repeatCount|effectedBellCount|effectedBells

Get current time

Function:
Get current time
Format:
/<getTime>
Parameters:
getTime: Api.
Return:
The current time of school bell system with format:
hour:minute:second day/month/year

Set new time

Function:

Set new time **Format:** **/<settime>/<second>/<minute>/<hour>/<date>/<month>/<year>** **Parameters:** **settime**: Api. **second**: secound **minute:** minute **hour:** hour **date:** date **month:** month **year:** year **Return:** **ok** if success, **ERROR** if failure.
Get the list bell
Function: Get the list bell **Format:** /<getBells> **Parameters:** **getBells**: Api. **Return:** all the current bells line by line each line is a bell with format: bellID|bellName
Add a new bell to system
Function: Add a new bell to system **Format:** /<addBell>/<bellID>/<bellName> **Parameters:** **addBell**: Api. **bellID**: ID of the bell **bellName:** Name of the bell **Return:** **ok** if success, **ERROR** if failure.
Remove a bell from system
Function: Remove a bell from system **Format:** /<removeBell>/<bellID> **Parameters:** **removeBell**: Api. **bellID**: ID of the bell **Return:** **ok** if success, **ERROR** if failure.

6. Data Structure

6.1. Database structure design

6.1.1. Table bell

Column	Type	Null	Unique	P/F Key	Description
id	INTEGER		x	Primary key	Id of timer.
name	TEXT	x			Name of timer.

6.1.2. Table alarm

Column	Type	Null	Unique	P/F Key	Description
bell_list	TEXT			Foreign key	List of bell in this alarm
repeat	INTEGER				Indicate is timer repeat or not.
day_list	INTEGER		x	Primary key	List of day in week
hour	INTEGER		x		Hours of timer.
minute	INTEGER		x		Minutes of timer.

V. System Implement & Test

In this section:

- Defining how testing would be carried out, the test strategy implementation of project ASBS (in section 1. Test approach).
- Giving screenshots which show the interface from user's perspective of ASBS Mobile Application (in section 2. Screen shots).
- Giving Master Test Plan of ASBS which gathers the information to plan and control the test effort for testing ASBS project. It describes the approach to testing the software and will be the top-level plan used by testers to direct the test effort (in section 3. Test plan).
- Providing test cases based on Test plan. In this document, only test cases for UI Test and functional test at level System test will be provided (in section 4. Test case).
- Giving checklists as the result of Static testing (in section 5. Check list).
- The Test log will be provided (in section 6. Other material).

1. Test Approach

ASBS is an embedded which was developed base on Iterative and incremental process model (Reference to ASBS Report 2: Part II. Project Organization). Testing does not exist in isolation; testing activities are related to software development activities (ISTQB, 2011). Thus, all testing activities in ASBS project must relate to the Iterative and incremental project model. The following figure gives a view into how testing activities can interact with the iterative model.

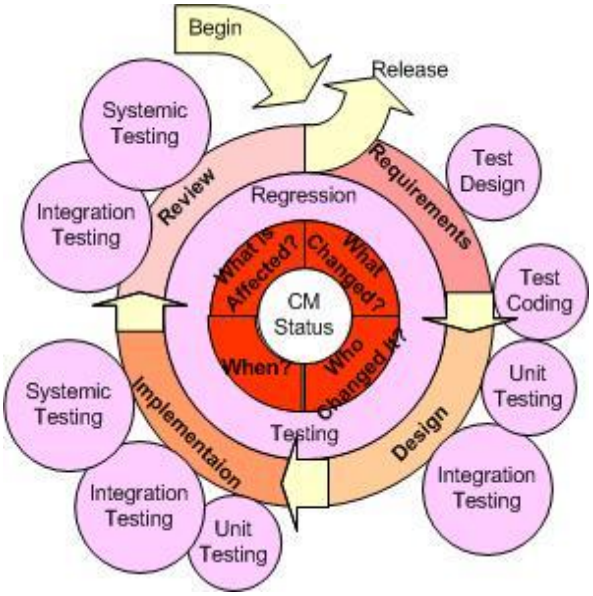


Figure 1:Testing activity in the Iterative model

However, because of the characteristic of the system and little changing in system, Regression test won't be done the scope of ASBS project.

Component test (a.k.a. Unit test):

Test basis:

- Component requirement
- Detailed Design
- Code

Typical test object:

- Components
- Programs
- Data conversion
- Database modules

Component testing (also known as unit, module or program testing) searches for defects in, and verifies the functioning of, software modules, programs, objects, classes, etc., that are separately testable. It may be done in isolation from the rest of the system, depending

on the context of the development life cycle and the system. Stubs, drivers and simulators may be used.

Component testing may include testing of functionality and specific non-functional characteristics, such as resource-behavior (e.g., searching for memory leaks) or robustness testing, as well as structural testing (e. g., decision coverage). Test cases are derived from work products such as a specification of the component, the software design or the data model.

Typically, component testing occurs with access to the code being tested and with the support of a development environment, such as a unit test framework or debugging tool. In practice, component testing usually involves the programmer who wrote the co de. Defects are typically fixed as soon as they are found, without formally managing these defects.

One approach to component testing is to prepare and automate test cases before coding. This is called a test-first app roach or test-driven development. This approach is highly iterative and is based o n cycles of developing test cases, then building and integrating small pieces of code, and executing the component tests correcting any issues and iterating until they pass.

Integration test

Test basis:

- Software and system design
- Architecture
- Workflows
- Use cases

Typical test object:

- Subsystems
- Database implementation
- Infrastructure
- Interfaces
- System configuration and configuration data

Integration testing tests interfaces between components, interactions with different parts of a system, such as the operating system, file system and hardware, and interfaces between systems.

There may be more than one level of integration testing and it may be carried out on test objects of varying size as follows:

1. Component integration testing tests the interactions between software components and is done after component testing.
2. System integration testing tests the inter actions between different systems or between hard ware and software and may be done after system testing. In this case, the developing organization may control only one side of the interface. This might be considered as a risk.

Business processes implemented as workflows may involve a series of systems. Cross-platform issues may be significant.

The greater the scope of integration, the more difficult it becomes to isolate defects to a specific component or system, which may lead to increased risk and addition al time for troubleshooting.

Systematic integration strategies may be based on the system architecture (such as top-down and bottom-up), function al tasks, transaction processing sequences, or some other aspect of the system or components. In order to ease fault isolation and detect defects early, integration should normally be incremental rather than “big bang”.

Testing of specific non-functional characteristics (e.g., performance) may be included in integration testing as well as functional testing.

At each stage of integration, testers concentrate solely on the integration itself. For example, if they are integrating module A with module B they are interested in testing the communication between the modules, not the functionality of the individual module as that was done during component testing. Both functional and structural approaches may b e used.

Ideally, testers should understand the architecture and influence integration planning. If integration tests are planned before components or systems are built, those components can be built in t he order required for most efficient testing.

System test

Test basis:

- System and software requirement specification
- Use cases
- Functional specification
- Risk analysis report

Typical test object:

- System, user and operational manuals
- System configuration and configuration data

System testing is concerned with the behavior of a whole system/pro duct. The testing scope shall be clearly addressed in the Master and/or Level Test Plan for that test level.

In system testing, the test environment should correspond to the final target or production environment as much as possible in order to minimize the risk of environment-specific failures not being found in testing.

System testing may include tests based on risks and/or on requirements specifications, business processes, use cases, or other high level text descriptions or models of system behavior, interactions with the operating system, and system resources.

System testing should investigate functional and non-functional requirements of the system, and data quality characteristics. Testers also need to deal with incomplete or undocumented requirements. System testing of functional requirements starts by using the most appropriate specification-based (black-box) techniques for the aspect of the system to be tested. For example, a decision table may be created for combinations of effects described in business rules.

2. Screen Shots

(Reference to Prototype)

3. Test Plan

3.1. Feature to be tested

- Turn on/off bell remote using mobile application.
- Connect smart phone to modem
- View all bell connect with connected Arduino
- View all schedule
- Add new schedule
- Delete selected schedule
- Delete all schedule
- Change name of bell
- Authentication

3.2. Feature not to be tested

- Change password

3.3. Test strategy

Functional testing

Test Objective	Ensure proper target-of-test functionality, including navigation, data entry, processing, and retrieval. Verify system functional requirements.
Technique	Execute each use case, use-case flow, or function, using valid and invalid data, to verify the following: The expected results occur when valid data is used. The appropriate error or warning messages are displayed when invalid data is used.

	Each business rule is properly applied.
Completion Criteria	All of the following are successfully tested: - All key use-case scenarios - All key features All identified defects have been addressed.
Special Consideration	N/A.

User Interface testing

Test Objective	Verify the following: Navigation through the target-of-test properly reflects business functions and requirements, including window-to-window, field-to-field, and use of access methods (tab keys, mouse movements, accelerator keys) Window objects and characteristics, such as menus, size, position, state, and focus conform to standards.
Technique	Create or modify tests for each window to verify proper navigation and object states for each application window and objects.
Completion Criteria	Each window successfully verified to remain consistent with benchmark version or within acceptable standard
Special Consideration	N/A.

Testing environment

Base Hardware		
Resource	**Quantity**	**Name and Type**
Laptop	2	
Android Smart Phone	1	
Wi-Fi embedded module	1	HI-LINK HLK-RM04
Bell	1 2	Super-bell Mini-speaker
Wireless Access Point	1	TP-LINK Access Point
Base Software		
Resource	**Quantity**	**Name and Type**
Window 7	1	Operating system
Android OS	1	Android OS 5.1
Android Development Tools (ADT)	1	ADT 36.6.3
Arduino IDE	1	Arduino 1.0.5

4. Test Case

(Reference to file ASBS_Testcase_final.xslm)

5. Other material

5.1. UI Test result

Features	Passed	Failed	Untested	Test Cases
Total	**19**	**0**	**0**	**19**
Turn on/off bell remote using mobile application	2	0	0	2
Connect smart phone to modem	1	0	0	1
View all bell connect with connected Arduino	1	0	0	1
View all schedule	1	0	0	1
Add new schedule	1	0	0	1
Delete selected schedule	2	0	0	2
Delete all schedule	1	0	0	1
Change name of bell	2	0	0	2
Authentication	2	0	0	2

5.2. Unit test firmware result

Features	Passed	Failed	Untested	Test Cases
Total	18	**0**	**0**	18
Setup	3	0	0	3
Create new bell	5	0	0	5
Reset interupt	1	0	0	1
Loop	8	0	0	8
Alarm check	1	0	0	1

5.3. System test result

Features	Passed	Failed	Untested	Test Cases
Total	48	**0**	**0**	48
Turn on/off bell remote using mobile application	6	0	0	6
Connect smart phone to modem	4	0	0	4
View all bell connect with connected	2	0	0	2

Arduino				
View all schedule	6	0	0	6
Add new schedule	8	0	0	8
Delete selected schedule	7	0	0	7
Delete all schedule	4	0	0	4
Change name of bell	5	0	0	5
Authentication	6	0	0	6

VI. Conclution

This section is the final document produced for the project. In this section, achieved original objectives will be provided, lessons learned will be outlined and formally close the project.

Automatic school bell system project has achieved the following original objectives:

- Created smart belling system.
- Provided a belling system consist of smart bell.
- Developed a mobile application to control the belling system.
- Implemented all main function of the system.
- All deliverables were delivered and approved on time.

Lessons learned, what did not go well:

Lesson No	Lesson Description	Suggested future action	Project Impact
1	Team member didn't focus on completing assigned task	Preparing communication plan before kick-off project. Providing encouragement mechanism with bonus and penalty.	High
2	System architecture is not stable	System architecture design need to be approved by supervisor before implementing.	Medium

VII. System User's Manual (SUM)

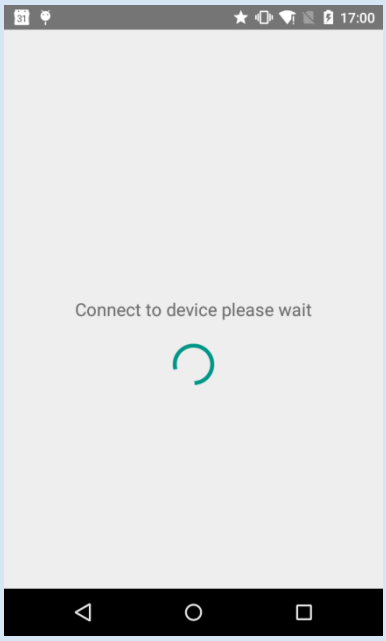
In this section:

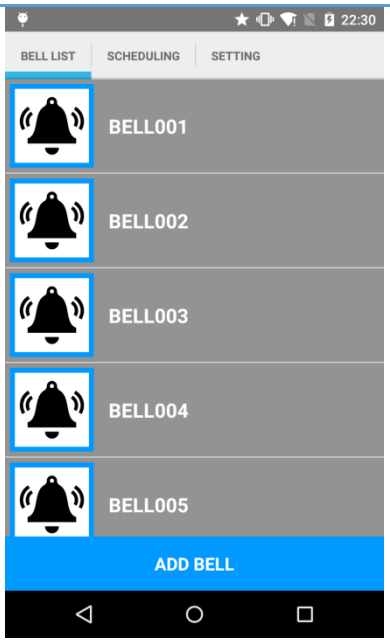
- Providing Installation Guide to show how to set up ASBS system at home:
  - Setup hardware: set up ASBS belling system consists of smart bell.
  - Setup software: set up ASBS mobile application on the user's smart phone.
- Providing User's Guide to guide users how to use ASBS mobile application to control ASBS belling system.

(In scope of a User manual, from this part, "You" pronoun will be used that mean "the user" – who buys and uses ASBS Belling system)

1. Android Application

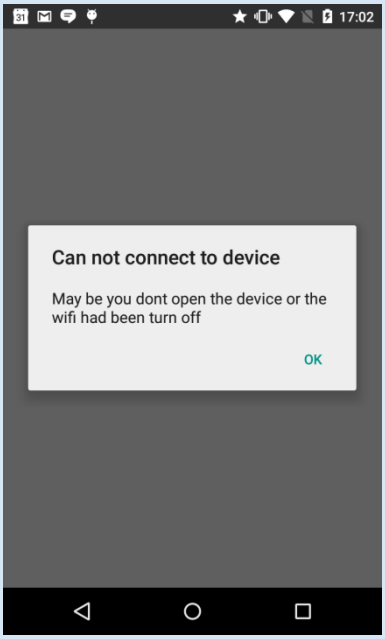
1.1. Detect and connect to bell system

Detect and connect to bell system
Step 1: Click on ASBS icon from list app. Loading screen will showing. 
Step 2: Waiting for Connector Application finding ASBS Belling system. If finding success, showing list bell screen.



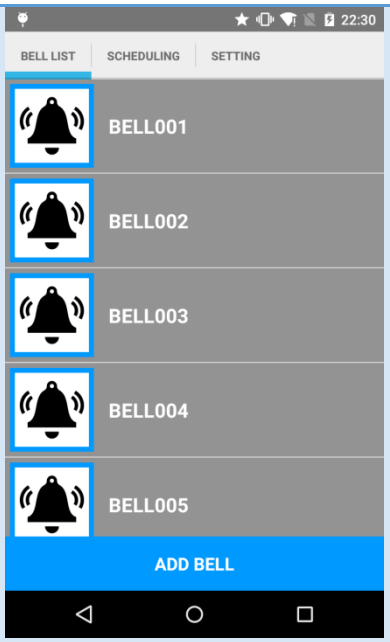
Step 3:

If can find belling system, a error dialog will be showed.



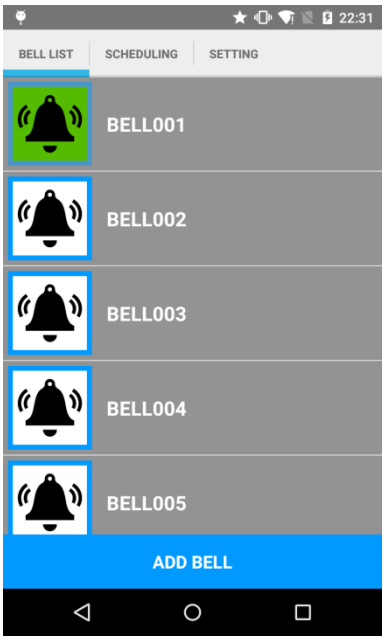
1.2. Turn on bell

Turn on bell
Step 1: Click on bell icon.



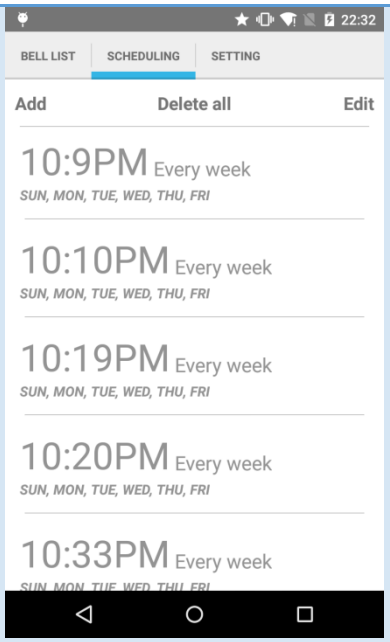
Step 2:

If the icon change to green color, the bellwill ringing.



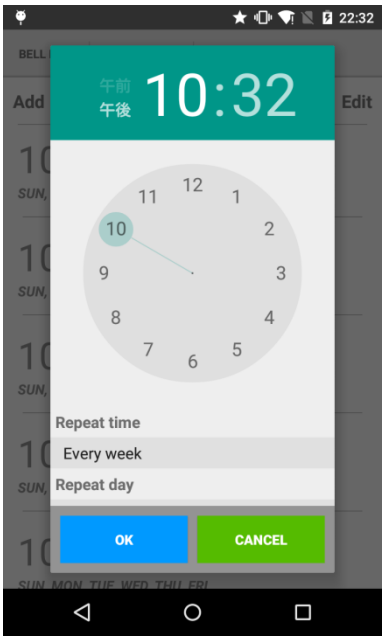
1.3. Update schedule

Update the schedule
Step 1: User swipe to left. Schedule screen will showing up



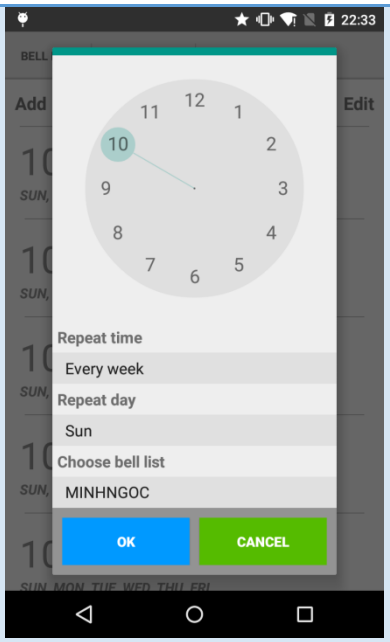
Step 2:

User tap on Add button. Add dialog will be showing up.



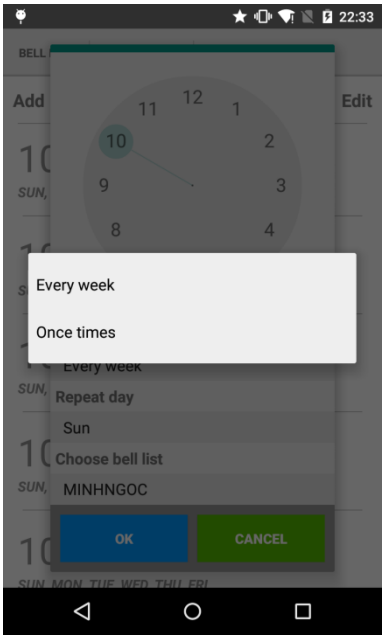
Step 3:

User could scroll down for more information.



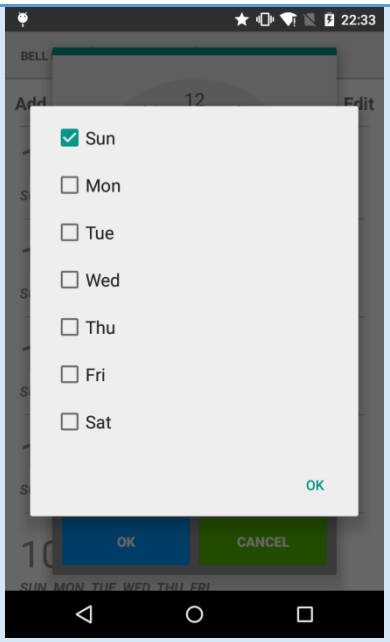
Step 4:

User click on repeat time for choosing repeat time of the bell.



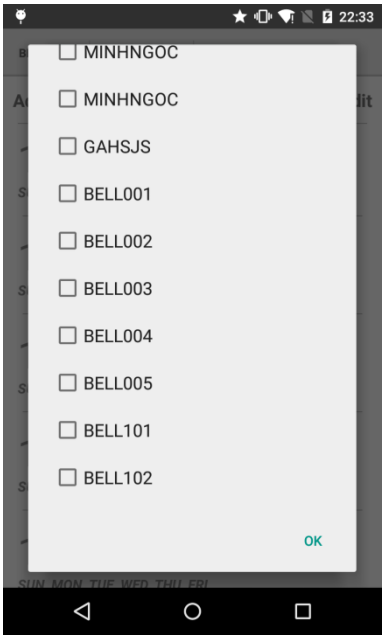
Step 5:

User click on repeat day for choosing repeat day of the bell.



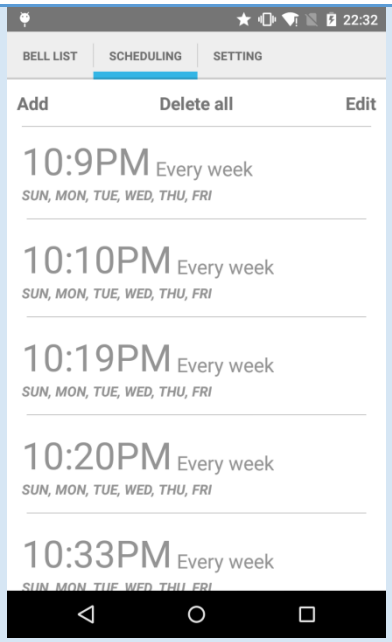
Step 6:

User click on bell lisst for view list of the bell.

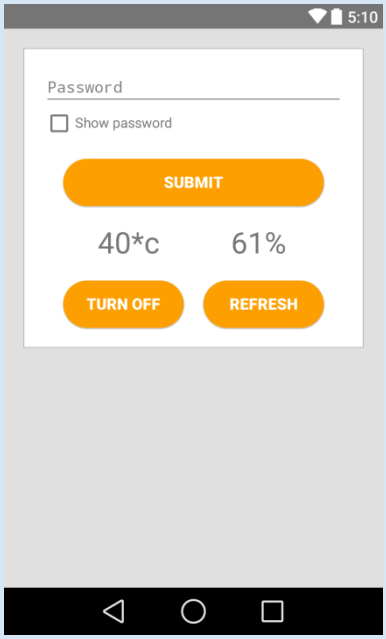


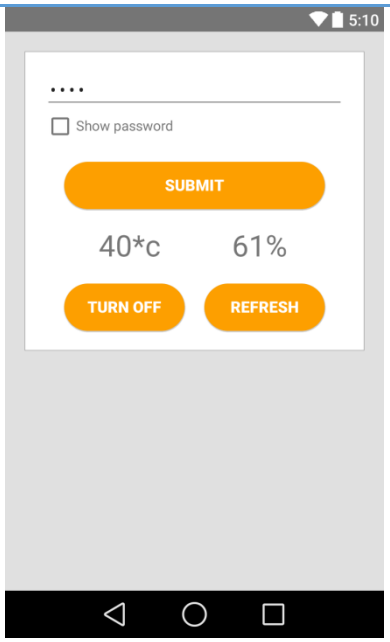
Step 7:

User click on “OK” button. The lisst will be reloaded.



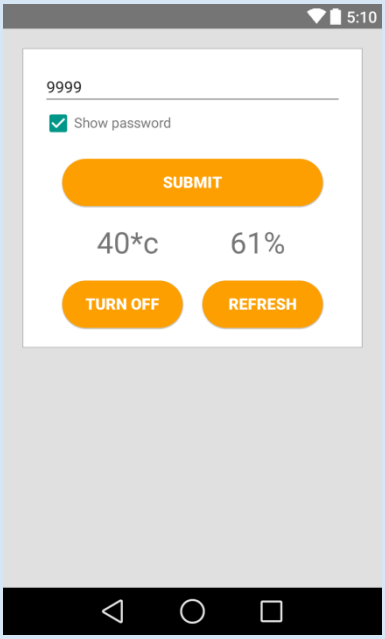
1.4. Change authentication password

Change authentication password
Step 1: User swipe to the left. Setting screen will showing up 
Step 2: User input the new password.



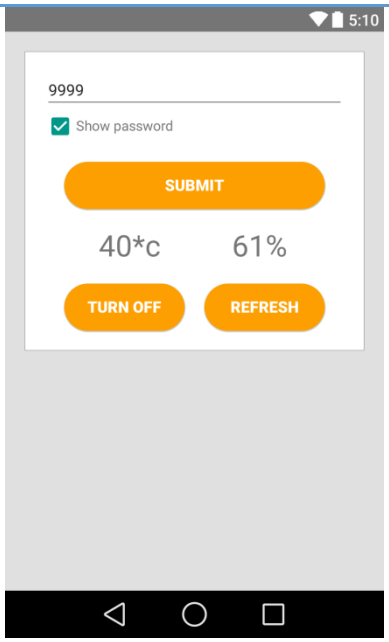
Step 3:

User could check on “show password” check box to show the current password in the edit text view.



Step 4:

The edit text box also support user to input password by seeing them.



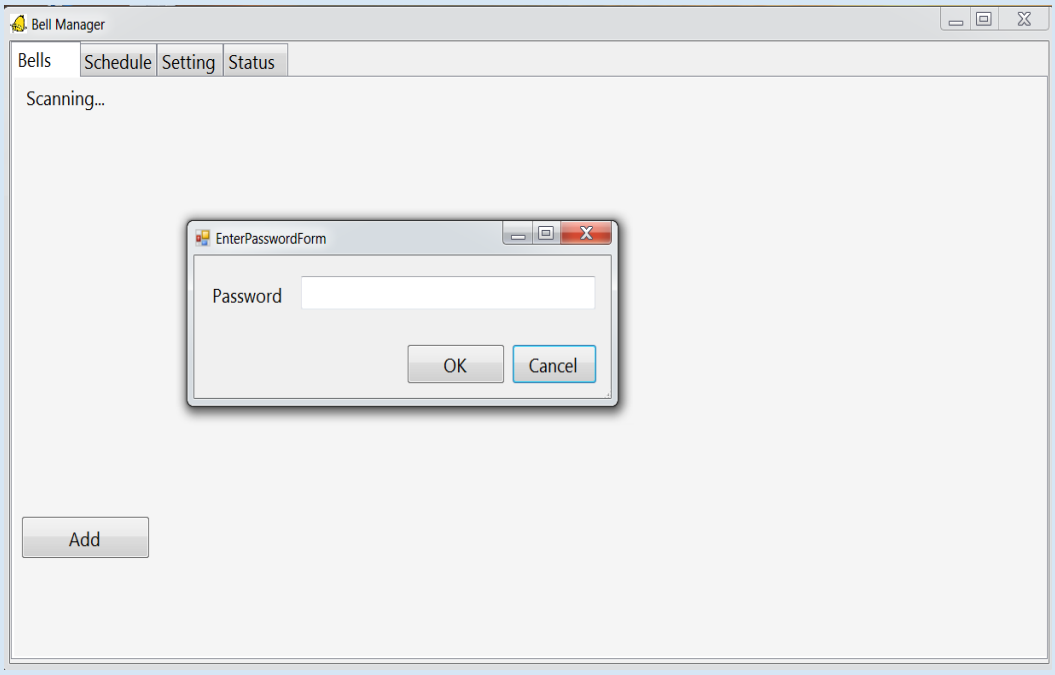
2. PC Application

2.1. Detect and connect to bell system

Detect and connect to bell system

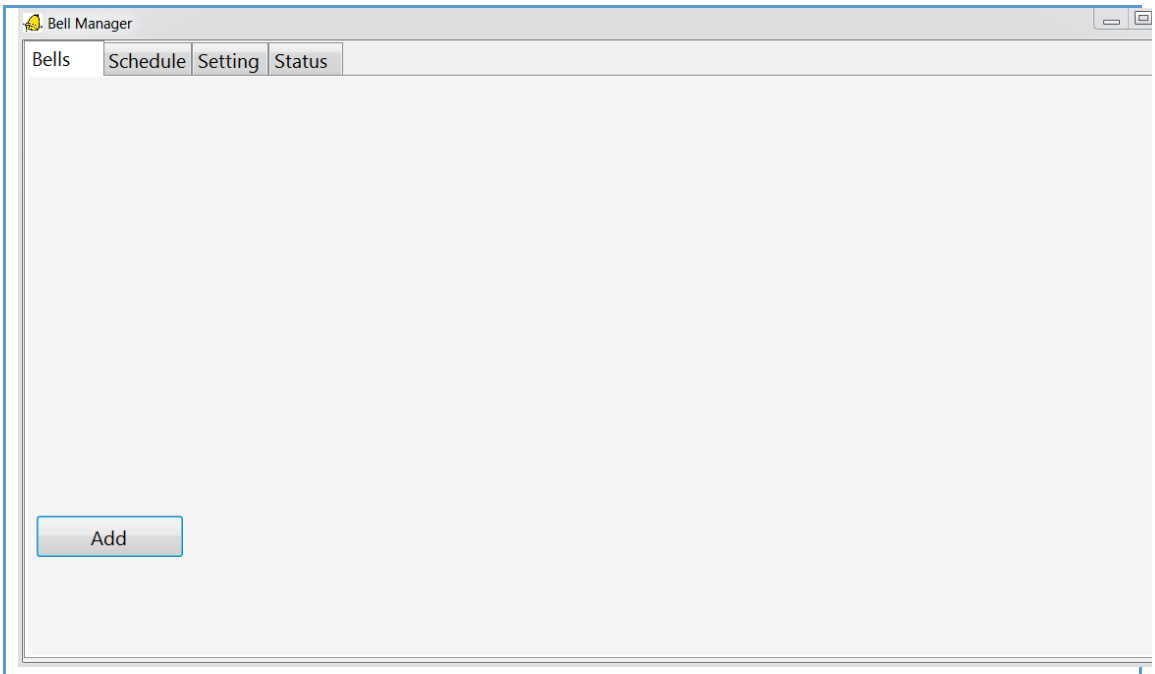
Step 1:

Click on ASBS icon in PC. Loading screen will showing and enter password.



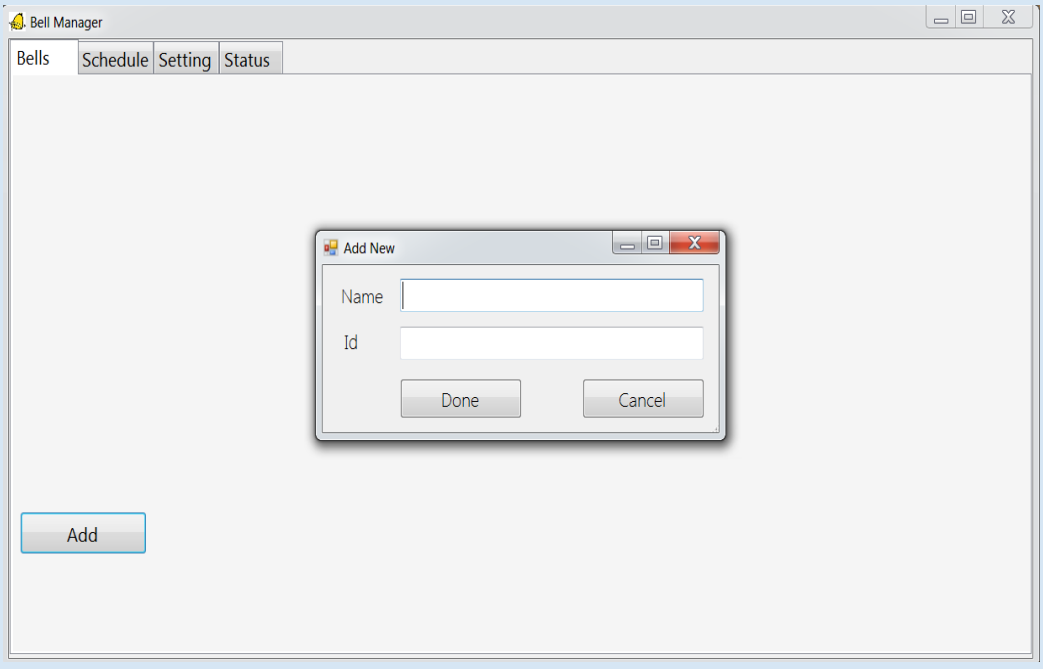
Step 2:

Waiting for Connector Application finding ASBS Belling system. If correct password , showing list bell screen.



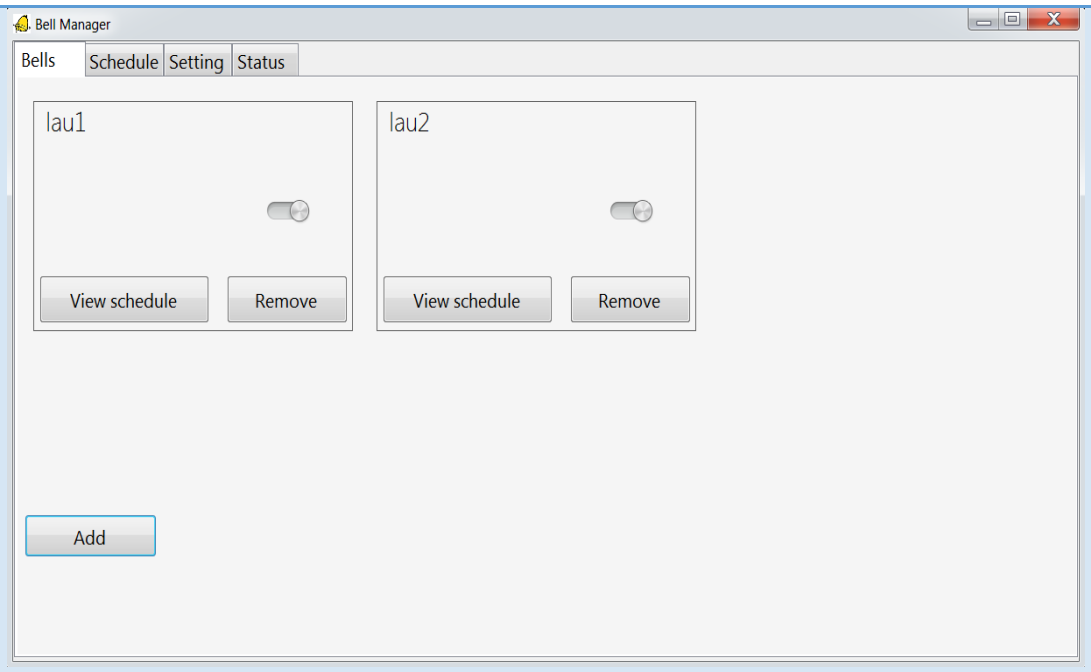
Step 3:

First , In Bells status . User tap on Add button to add bell



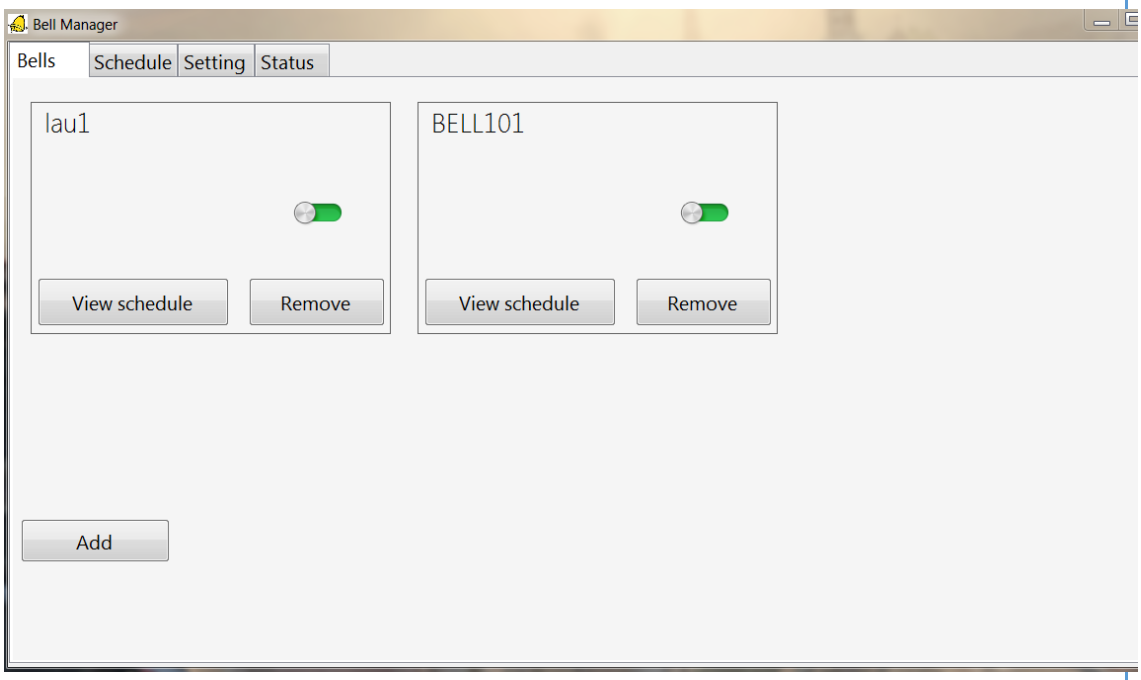
2.2. Turn on bell

Turn on bell
Step 1: Click on bell icon.



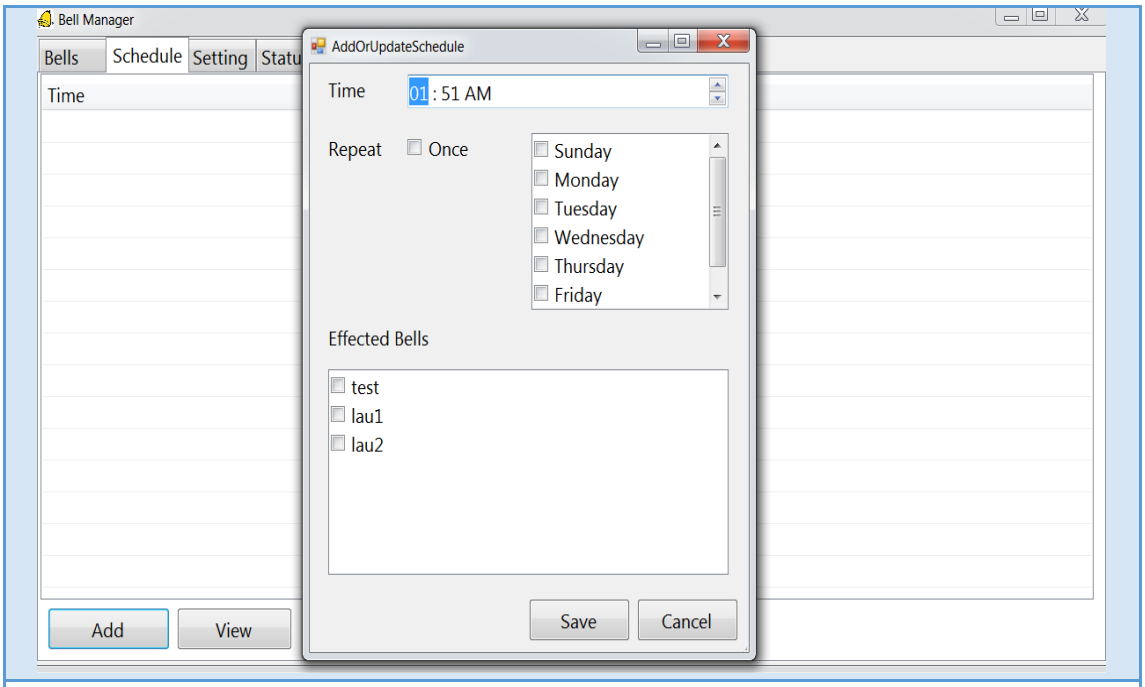
Step 2:

If the icon change to green color, the bellwill ringing.



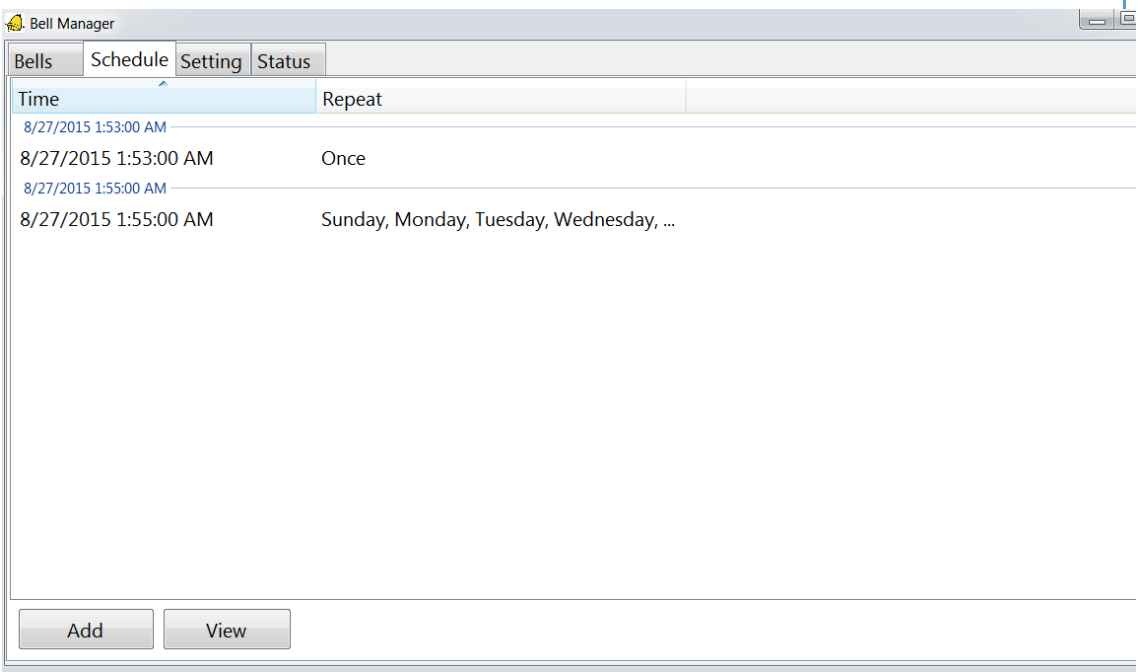
2.3. Update schedule

Update the schedule
Step 1: In next Status. Schedule screen will showing up when tap button Add



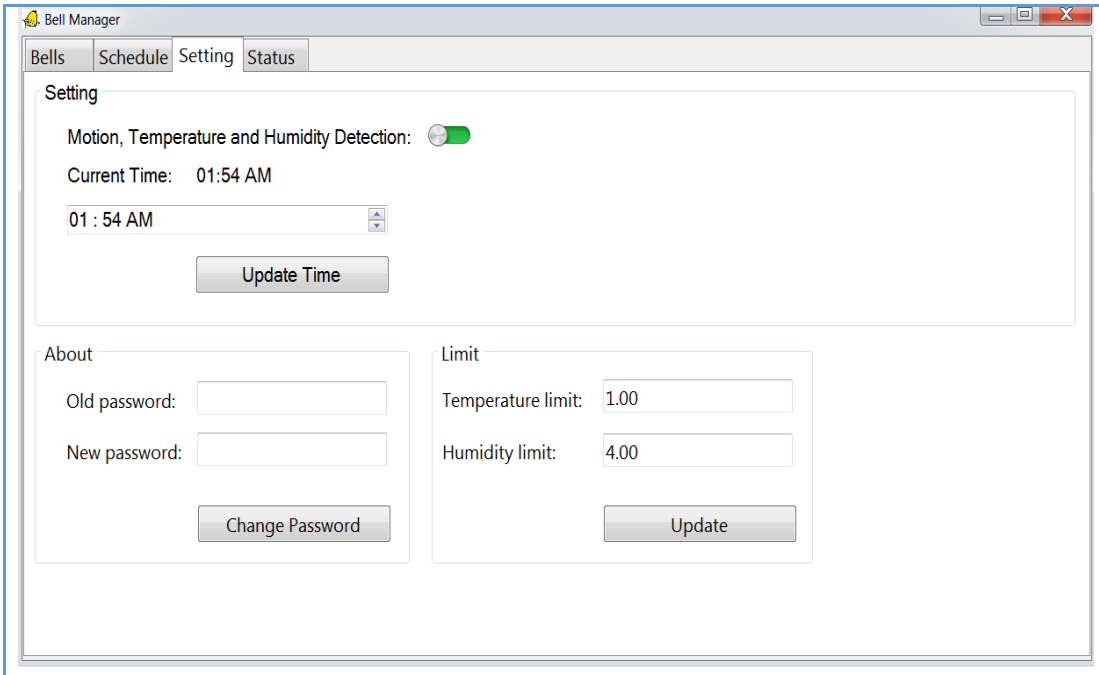
Step 2:

User click on "OK" button. The lisst will be reloaded.



2.4. Setting

In setting screen is showing up.



User can update time , change password , and update limit Temperature , humiditylimit and turn on or off Motion , temperature , Humidity.

2.5. Status

Step 1:

In Status. Screen is showing Weather info